UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA MECÁNICA

ÁREA DE INGENIERÍA DE ORGANIZACIÓN

PROYECTO FIN DE CARRERA

HERRAMIENTA INFORMÁTICA PARA MODELADO FLEXIBLE DE PROBLEMAS DE ASIGNACIÓN DE INFRAESTRUCTURAS

ALUMNO: FERNANDO JOSÉ CARRASCO PÉREZ

TUTOR: MIGUEL GUTIÉRREZ FERNÁNDEZ

SEPTIEMBRE DE 2011

Título: Herramienta informática para modelado flexible de problemas de asignación de infraestructuras

Autor: Fernando José Carrasco Pérez

Tutor: Miguel Gutiérrez Fernández

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 19 de Septiembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

SECRETARIO VOCAL PRESIDENTE

AGRADECIMIENTOS

A mi familia, por su apoyo y confianza en todos los retos a los que me he enfrentado.

A mis amigos, que comparten mi ilusión por superarse cada día.

A todos mis profesores, por el enorme valor que tiene su labor.

A Miguel Gutiérrez, por su ayuda, tiempo y consejos, sin los cuales no hubiera sido posible realizar este proyecto.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1

INTRODUCCIÓN

En la primera sección de este capítulo se exponen los antecedentes al presente proyecto, es decir, cómo se gestó la idea de su realización. Posteriormente se enuncia el objetivo que se persigue, se presenta la estructura que sigue este documento y, por último, se muestra la planificación y presupuesto del desarrollo de la herramienta informática y su documentación.

1.1 ANTECEDENTES

El presente proyecto se enmarca en una línea de investigación de la que participa el Área de Ingeniería de Organización. En concreto, entre 2006 y 2009 con el proyecto denominado *Sistema avanzado de ayuda a la toma de decisiones para la gestión hotelera*, y desde 2009 hasta 2012 a través del proyecto *Optimización de la asignación de infraestructuras de servicios mediante simulación – sectores hotelero y sanitario*.

En Febrero de 2010 se entró en contacto con Alfonso Durán, Catedrático del Área de Ingeniería de Organización y responsable de los citados proyectos. Se inició una primera relación de colaboración en el marco de un trabajo dirigido. Las labores realizadas fueron fundamentalmente de documentación y modelado en el campo de los problemas de asignación de infraestructuras.

A principios de Diciembre de 2010 se concretó la posibilidad de abordar la realización de este proyecto fin de carrera. Se resolvió que versaría sobre el diseño y programación de una herramienta informática que asistiese al modelado flexible de problemas de asignación de infraestructuras. El tema elegido me permitiría aprovechar las nociones básicas adquiridas en el campo del Revenue Management durante el trabajo dirigido, así como poner en práctica mis habilidades de programación e iniciarme en el uso de herramientas de desarrollo nuevas para mí, como UML, IDEF1X y Visual Basic, de las cuales se hablará en el siguiente capítulo.

En el párrafo anterior se ha mencionado un concepto, el de Revenue Management, que merece ser explicado con detenimiento. En el entorno empresarial actual, caracterizado por una fuerte competitividad, además de la progresiva terciarización de la economía, las compañías que mejor satisfacen las necesidades de sus clientes son las que consiguen una mayor rentabilidad, asegurándose de esta forma su supervivencia. El valor percibido por el cliente es una medida excelente del éxito de los servicios de una empresa. En la mayor parte de los casos, la competencia creciente y la saturación de los mercados presionan a los precios a la baja, de tal forma que las empresas entienden que la reducción de los costes resulta ser la única forma de asegurar sus márgenes y por ende su viabilidad. Sin embargo, existe otra perspectiva que se complementa con la anterior: atacar el problema desde el punto de vista del incremento de los ingresos, lo que contribuye al mantenimiento o mejora de la rentabilidad.

Las compañías de servicios que a corto y medio plazo cuentan con unas infraestructuras dadas, cuyo potencial de generación de ingresos es desaprovechado si no se utilizan en el tiempo y lugar adecuados, tienen una estructura de costes en la que

predominan los de carácter fijo. Estas empresas pueden ver incrementados sus ingresos mediante una asignación más eficiente de sus recursos. Para ello, pueden utilizarse técnicas de Revenue Management, también conocido como Yield Management o Gestión del Ingreso. De forma general, el Revenue Management consiste en reaccionar o anticiparse al comportamiento del consumidor, de tal forma que el ingreso es optimizado cuando se logra que cada cliente desembolse el máximo precio que está dispuesto a pagar. Implica la utilización de modelos dinámicos de predicción, la asignación de recursos o servicios perecederos a los diversos segmentos de clientes, canales y categorías de precios, así como el precio asociado a cada categoría.

El Revenue Management ha estado asociado a las aerolíneas desde su nacimiento en los años ochenta, que fueron las primeras en utilizarlo a gran escala. La desregularización del sector unida al progreso en el campo de los sistemas de información permitió a estas compañías articular sus políticas de precios de forma más inteligente teniendo en cuenta la previsión de la demanda para obtener el máximo partido de la limitada capacidad de sus aviones en cada ruta y horario. (Zeni, 2001). El éxito de las aerolíneas que comprendieron el potencial del Revenue Management e invirtieron en la tecnología necesaria para su correcta aplicación, atrajo la atención de otros sectores empresariales. Uno de ellos fue el hotelero, cuya naturaleza es totalmente compatible con la aplicación de estas herramientas.

La aplicación del Revenue Management se realiza con la asistencia de Sistemas de Soporte a la Decisión (Decision Support Systems, DSS de aquí en adelante). Los DSS son sistemas de información que proporcionan apoyo a las actividades que comportan toma de decisiones de gestión empresarial (Turban, 2004). Los DSS surgen como resultado de la fusión de dos líneas de trabajo distintas a comienzos de la década de los 1970s: los Sistemas de Información de Gestión (Management Information Systems, MIS) y la Investigación Operativa (Silver, 1991). La arquitectura básica de un DSS se articula en torno a tres componentes básicos: una base de datos, una base de modelos y un sistema software encargado de la gestión de la base de datos, de la base de modelos y de la interfaz de usuario (Sprague y Carlson, 1982).

La herramienta cuyo desarrollo se aborda en el presente proyecto constituye un primer paso hacia la obtención de un DSS. Se corresponde con la base de modelos y una parte de la interfaz de usuario. Pretende modelar un subconjunto de problemas de Revenue Management, en concreto el de aquellos que se basan en la asignación de infraestructuras a diferentes segmentos o agrupaciones de clientes, que acceden por distintos canales con el fin de reservarlas para su uso. Dichas infraestructuras tienen una capacidad limitada y pierden su potencial de generar valor por su uso todo el tiempo que permanecen vacantes.

1.2 OBJETIVO

Como se ha expuesto en la sección anterior, la herramienta pretende ser un asistente de definición para un determinado subconjunto de problemas de asignación de infraestructuras. En concreto, se trata de los problemas en los que se produce un acceso de un cliente por un canal para solicitar una reserva de una infraestructura. Se dice que es un subconjunto porque no estamos en disposición de afirmar que todos los problemas de asignación de infraestructuras son de esa naturaleza.

Así pues, el objetivo del proyecto se enuncia de la siguiente forma:

Desarrollar una herramienta informática fácilmente expandible que permita modelar de forma flexible, rápida e intuitiva problemas de asignación de infraestructuras a clientes, pertenecientes a un segmento o agrupación, que acceden por distintos canales para realizar una reserva.

El software a desarrollar debe cumplir una serie de requisitos para ser coherente con el objetivo propuesto:

- Debe fundamentarse en un modelo conceptual sólido que admita futuras expansiones sin perder su significado. En ese sentido, se cuenta con un modelo conceptual de partida (Gutiérrez y Durán, 2011), que se explicará en el segundo capítulo del documento.

- Para que una eventual expansión pueda ser sencilla, la arquitectura de la aplicación debe contar con dos elementos: la capa de datos y la interfaz gráfica de usuario. La justificación es doble. Por un lado permite que en un momento dado pueda reutilizarse gran parte del diseño para implementar la aplicación en otra plataforma. Por otro lado, al asegurar la independencia de aplicación y datos, se puede asegurar una cierta estanqueidad entre los distintos módulos que conforman la aplicación y facilitar el desarrollo de una expansión de la funcionalidad de la misma en caso de que se considere oportuno.
- Respecto a la capa de datos, mencionada en el requisito anterior, es necesaria una metodología para llevar a cabo la traducción del modelo conceptual a modelo de datos.
- Con el fin de que la definición de los problemas sea flexible, rápida e intuitiva, se tiene que abordar el diseño e implementación de la interfaz gráfica de usuario desde una óptica muy general, que dote al usuario de la capacidad de modificar numerosos aspectos. Como consecuencia, la complejidad de la aplicación puede crecer más de lo deseable. Así pues, la solución elegida tendrá que permitir al usuario familiarizarse fácilmente con el software así como completar un modelo de proceso o una instancia de proceso en un tiempo razonable manteniendo la flexibilidad. Modelo de proceso e instancia de proceso son conceptos que se explican en el segundo capítulo.

1.3 ESTRUCTURA DEL DOCUMENTO

El presente documento se estructura en nueve capítulos que pueden agruparse en tres partes. La primera parte incluye un capítulo de perspectiva general del proyecto a modo de introducción y otro de marco teórico. La segunda parte, que conforma el núcleo del documento, incluye los capítulos que detallan el diseño e implementación de los diferentes módulos que integran la herramienta informática. Por último, la tercera parte aglutina un ejemplo de utilización del software en la definición de un problema de asignación de infraestructuras, un capítulo de conclusiones y sugerencias acerca de trabajos futuros y la bibliografía utilizada en la realización del proyecto.

A pesar de que la estructura del documento es lineal y sigue una progresión lógica para la elaboración de la herramienta por un motivo de claridad expositiva, las diferentes etapas del proceso de diseño e implementación la herramienta informática no se han seguido de manera estrictamente ordenada, como ocurre con frecuencia en todo proyecto de desarrollo de software.

El primer capítulo del documento es la *Introducción*. Se trata de una guía en la que se abordan los antecedentes en la línea de investigación del proyecto fin de carrera, los objetivos del mismo, la estructura que sigue el presente documento y la planificación y presupuesto para la creación de la herramienta informática y su documentación.

A continuación, en el segundo capítulo, denominado *Marco Teórico*, se expone cuál es el modelo de partida para la elaboración del diseño conceptual de la herramienta, cuáles son las líneas generales del diseño de la herramienta y qué herramientas de desarrollo se han utilizado.

Los siguientes cuatro capítulos contienen la explicación del diseño conceptual, el modelo de datos, el diseño de la aplicación y su implementación, todo ello referido al módulo del software mencionado en su título: *Módulo de definición de Procesos de Negocio*, *Módulo de definición de Atributos*, *Módulo de definición de Estados* y *Módulo de definición de la Función Objetivo*.

El documento también incluye un capítulo de *Ejemplo de uso de la herramienta*, que muestra algunas de las posibilidades que ofrece en materia de definición de problemas de asignación de infraestructuras.

Las *Conclusiones* extraídas del desarrollo de la herramienta informática que nos ocupa se presentan en el penúltimo capítulo, incluyendo algunas ideas sobre los próximos pasos que pueden seguirse para complementarla o mejorarla.

El cierre del documento se produce con la *Bibliografía*. Es un listado de los libros utilizados en la creación de la herramienta y su documentación, ajustándose a la ISO 690-1987, que establece criterios internacionalmente aceptados en la elaboración de referencias bibliográficas.

1.4 PLANIFICACIÓN Y PRESUPUESTO

La realización del proyecto ha durado algo más de nueve meses. A continuación, en la figura 1-1, se presenta un diagrama de Gantt que muestra las diferentes tareas y subtareas que ha sido necesario completar para finalizar el proyecto. Cabe advertir, como ya se hizo en la sección anterior, que, como en la práctica totalidad de los proyectos de software, el proceso no ha sido tan lineal ni las tareas se han abordado con la estanqueidad que se muestra en el diagrama. Por ello, se ha intentado elaborar un diagrama lo más realista posible sin incrementar la complejidad de forma desmesurada. El diagrama de Gantt se complementa con la tabla 1-1, que recoge las horas-hombre dedicadas a las tareas fundamentales.

Tarea	Horas - hombre totales
Estudio inicial	**80**
Aprendizaje de las herramientas de desarrollo	**192**
Desarrollo de la herramienta informática	**561**
Desarrollo del Módulo de definición de Procesos de Negocio	181
Desarrollo del Módulo de definición de Atributos	80
Desarrollo del Módulo de definición de Estados	150
Desarrollo del Módulo de definición de la Función Objetivo	150
Pruebas de uso de la herramienta informática	**50**
Elaboración del documento del proyecto	**155**
	1038

Tabla 1-1. Distribución de la carga de trabajo por tarea

En la tabla 1-2 se muestra cómo se distribuye la carga de trabajo en distintos períodos:

Inicio del período	Fin del período	Días	Horas-hombre diarias	Horas-hombre totales
06/12/2010	15/01/2011	40	2	80
02/02/2011	31/05/2011	118	4	472
20/06/2011	09/09/2011	81	6	486
				1038

Tabla 2-1. Distribución temporal de la carga de trabajo

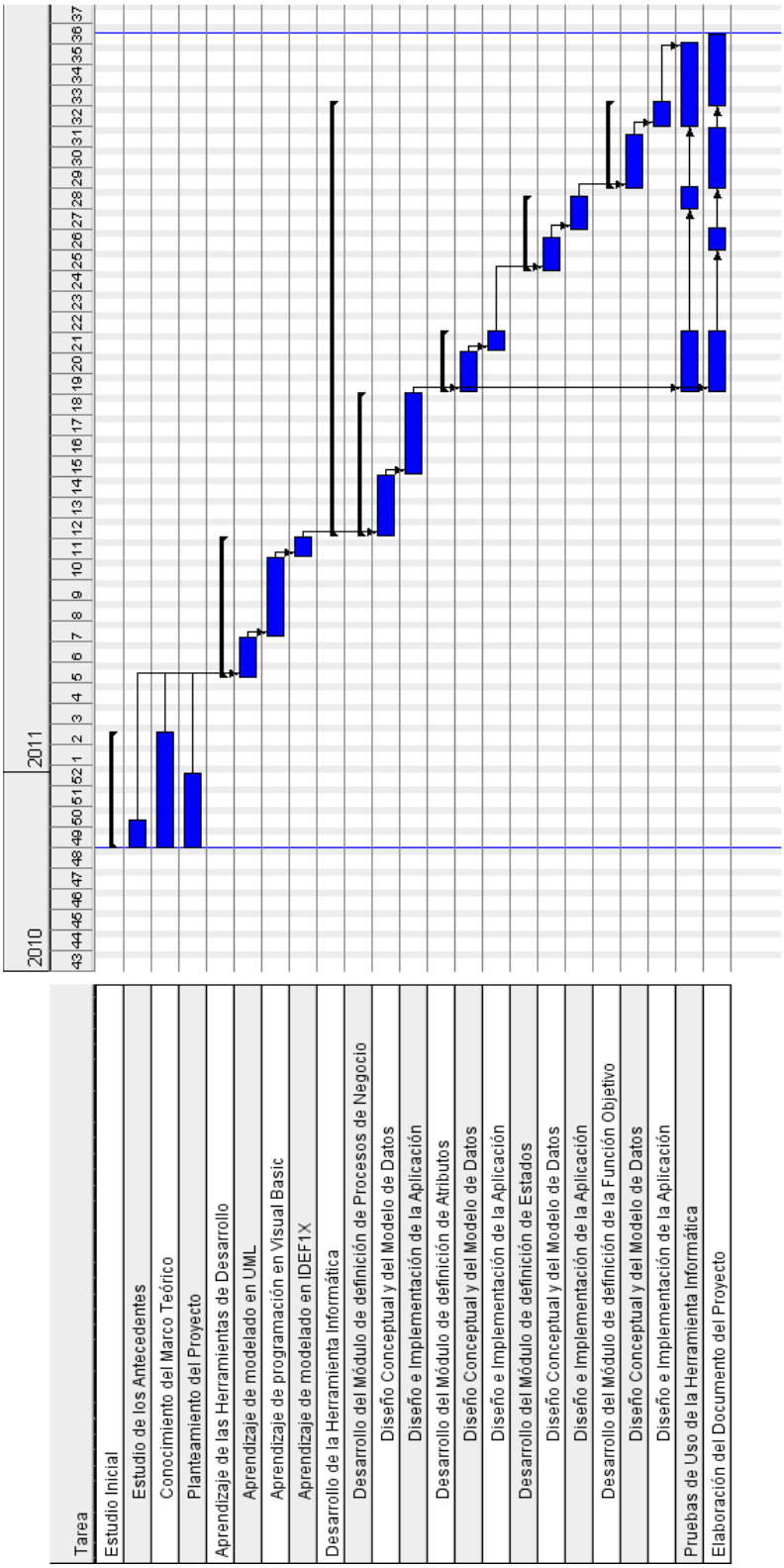


Figura 1-1. Diagrama de Gantt del proyecto

Si bien en el análisis de la carga de trabajo se incluye tanto el estudio inicial como el aprendizaje de las herramientas de desarrollo, hemos de tener en cuenta que si se tratase de un presupuesto profesional no se podrían introducir tales tareas como parte de la propuesta al cliente. En ese caso, la disminución de la carga de trabajo presupuestada se encontraría en torno al 25%. Si obviamos la consideración anterior y suponemos un coste de hora-hombre aproximado de 15 € para un ingeniero junior, el coste de personal se situaría en torno a los 15.570 €.

En lo que respecta a los medios utilizados para la realización del proyecto, el único necesario ha sido un ordenador Sony Vaio, modelo VGN-CS21S, con procesador Intel Core 2 Duo T6400 a 2.00 GHz, con 4 GB de memoria RAM y sistema operativo MS Windows Vista Home Premium.

En el caso de un presupuesto profesional, el coste debido a la utilización del ordenador no se imputaría por separado, sino que normalmente se tendría en cuenta como parte de los costes generales. Si pasamos por alto esta observación y suponemos que se le presenta al cliente este coste desglosado como parte del presupuesto, y además que la amortización del ordenador es lineal, con periodo de amortización de cinco años, valor residual nulo y un precio de mercado en el momento de compra de 1.000 €, el coste de amortización de los nueve meses de uso del ordenador se sitúa en 150 €.

Teniendo presentes todas las observaciones anteriores, así como las suposiciones que se han hecho, el coste total del proyecto se situaría en el entorno de los 15.720 €.

2

Marco Teórico

Como ya se explicó en la Introducción del documento, la estructura de este capítulo se articula en torno a tres núcleos fundamentales. Primero, se explica el modelo de partida en el cual se inspira el diseño conceptual de la herramienta. A continuación, se esbozan las líneas maestras del diseño de la herramienta, para finalizar con una presentación de las herramientas de desarrollo útiles para la realización del proyecto.

2.1 MODELO DE PARTIDA

El modelo original de definición de un proceso de negocio aparece pertenece al trabajo de investigación de Gutiérrez y Durán, de 2011, *Generic Model Base Design for Decision Support Systems in Revenue Management: Applications to Hotel and Health Care Sectors*. Se define un Modelo de Proceso de negocio (Business Process Model) como el proceso de asignación de los elementos que componen una infraestructura, para su uso presente o futuro. Si consideramos por ejemplo un negocio de alquiler de coches, el Modelo de Proceso de negocio incluye el diseño del proceso de reserva de los coches, teniendo en cuenta aspectos como la segmentación de la base de clientes, los diferentes canales a través de los cuales puede reservarse un automóvil, la fijación del precio de cada modelo y sus variaciones a lo largo del tiempo.

El modelado genérico se aborda considerando diferentes niveles de abstracción. Los primeros tres niveles, de mayor a menor nivel de abstracción, son los siguientes:

- El nivel del Metamodelo de Proceso de Negocio (Business Process Metamodel). El modelado genérico de procesos de negocio de asignación de infraestructuras permite relacionar los elementos básicos necesarios para que, a través de instanciaciones, puedan obtenerse distintos modelos de proceso de negocio.
- El nivel del Modelo de Proceso de Negocio (Business Process Model). Se corresponde con las instancias directas del nivel anterior.
- El nivel de la Instancia de Proceso de Negocio (Business Process Instance). Se trata de una instancia directa de un modelo de Proceso de negocio. Un mismo modelo de Proceso de negocio puede instanciarse varias veces, es decir, ser padre de varias Instancias de Proceso de negocio. Al completar una Instancia, función objetivo incluida, se está planteando un problema de asignación de infraestructuras, en el que algunos de los elementos del modelo serán parametrizados, mientras que otros actuarán como variables.

A modo de ejemplo, se puede establecer un paralelismo con un problema de optimización. En primer lugar, contamos con un metamodelo de funciones de optimización, que nos indica cómo se construye una función de optimización, cuáles son sus elementos y de qué formas pueden combinarse para dar lugar a distintas funciones. En el nivel inmediatamente inferior se encontrarían las funciones de optimización genéricas, como por ejemplo $f = i + j - k^2$, perteneciendo i, j y k al conjunto de los números naturales. Finalmente, en el tercer nivel tendríamos diferentes funciones en las que algunos de los elementos han sido parametrizados y otros actúan como variables de optimización, además de concretarse qué tipo de optimización se

desea (maximización o minimización). Por ejemplo, $Max\ f = 17 + j - 5^2$ ó $Min\ f = 2 + 10 - k^2$.

Para abordar la explicación de los niveles anteriores de una manera más efectiva, se utilizarán en los siguientes apartados de esta misma sección, así como en capítulos posteriores, diagramas de clase en lenguaje UML, una de las herramientas de desarrollo cuyo funcionamiento se expone en la tercera sección de este capítulo.

2.1.1 METAMODELO DE PROCESO DE NEGOCIO

El metamodelo propuesto, como modelo de modelos de Procesos de Negocio, pretende dar apoyo a la definición de una amplia base de éstos. Los elementos que forman parte del metamodelo, así como las relaciones entre ellos, están representados en la figura 2-1. A continuación, se explicará en cual es el contenido conceptual de cada uno de los elementos y cómo se asocian entre sí.

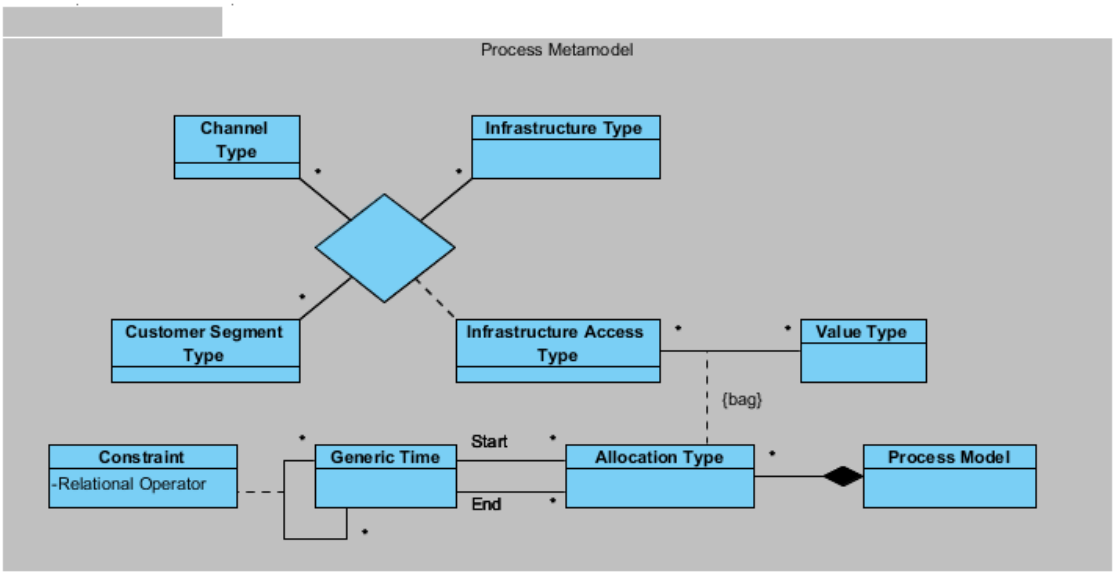


Figura 2-1. Diagrama de clase en UML del Process Metamodel en el modelo de partida

Un tipo de segmento de clientes (Customer Segment Type) es una forma de agrupar a los clientes basándose en una serie de características comunes. Todos los clientes que cuentan con unas determinadas características, pertenecen al mismo segmento. Estas características pueden ser geográficas, demográficas, conductuales, etc. Por ejemplo, muchas compañías segmentan a sus clientes en función de su lealtad a sus productos o servicios, lo que constituye un tipo de segmentación conductual. La segmentación de la base de clientes es una práctica muy útil en el desarrollo de la estrategia de marketing de una empresa. Le permite planificar y dirigir acciones comerciales específicas hacia aquellos segmentos que desea servir prioritariamente y

que son claves en el esquema de rentabilidad de la compañía, reforzando así su posición frente a la competencia y asegurando la sostenibilidad de su modelo de negocio. En Revenue Management, la segmentación es un concepto fundamental, pues el conocimiento que la empresa dispone de cada cliente es el que permite ofrecerle el producto o servicio al máximo precio que está dispuesto a pagar.

Otro elemento del metamodelo es el tipo de canal (Channel Type), que en nuestro caso concreto es la forma a través de la cual el cliente accede a la oferta de infraestructuras disponible. Ejemplos de tipos de canal son el teléfono o una página web para reservas. La oferta de infraestructuras suele incluir distintos tipos de infraestructura (Infrastructure Types). Por ejemplo, en el caso de un hotel, las habitaciones simples y las habitaciones dobles son tipos de infraestructura diferentes.

Conociendo los tres elementos anteriores, se puede definir un tipo de acceso a una infraestructura (Infrastructure Access Type) como la combinación de un tipo de segmento de cliente (Customer Segment Type) que accede a través de un tipo de canal (Channel Type) para que le sea asignado un tipo de infraestructura concreto (Infrastructure Type). En el caso más general, todos los tipos de segmentos de clientes podrían acceder a todos los tipos de infraestructura a través de todos los tipos de canal. Si añadimos restricciones al acceso, por ejemplo permitiendo tan sólo ciertos a tipos de segmento de cliente la reserva de un tipo de infraestructura determinado o eliminando la posibilidad de reservar algunos tipos de infraestructura a través de ciertos tipos de canal, estaremos ante un subconjunto de problemas de asignación de infraestructuras cuya definición puede abordarse con ayuda del metamodelo.

En el metamodelo también tienen cabida los tipos de valor (Value Type). En los problemas de asignación de infraestructuras orientados a fines exclusivamente comerciales, los diferentes tipos de valor serán los distintos tipos de tarifa: tarifa normal, reducida, de oferta, para jóvenes, de temporada alta, etc. Sin embargo, en ámbitos en los que el rédito económico no es el factor clave, como el sector sanitario, el tipo de valor puede referirse a un concepto más abstracto, como por ejemplo el beneficio social que reporta la asignación de una infraestructura a un cierto tipo de segmento de cliente.

Para continuar con la exposición, es necesario introducir el concepto de instanciación. Instanciar una clase es obtener un elemento más concreto a partir de un elemento abstracto. Es decir, se trata de la acción opuesta a la abstracción. Por ejemplo, existe la clase silla, como idea general de mueble que sirve de asiento a una sola persona. Sin embargo, cada una de las sillas que tenemos en nuestra casa, como objetos individuales y diferenciables, son instancias de la clase silla. De forma natural estamos instanciando en nuestro proceso de razonamiento cada vez que ponemos un ejemplo para ilustrar una idea.

Cada tipo de acceso a una infraestructura (Infrastructure Access Type) puede instanciarse de forma directa en tipos de acceso a una infraestructura concretos. A cada una de estas instanciaciones se le relaciona con un tipo de valor (Value Type) y dos tiempos genéricos (Generic Times). Esta combinación conforma la definición de un tipo

de asignación (Allocation Type). Así pues, un tipo de asignación es un tipo de acceso a una infraestructura, asociado a un tipo de valor, y que está vigente durante un intervalo de tiempo cuyos extremos inferior y superior son, respectivamente, su tiempo de comienzo genérico (Generic Start Time) y el tiempo de finalización genérico (Generic End Time). Estos tiempos se denominan como genéricos porque en este nivel no toman un valor concreto. Por ello se definen también de forma genérica las relaciones de comparación entre ellos, denominadas restricciones (Constraints), que se sirven de los operadores relacionales igual y menor o igual.

Un modelo de proceso de negocio (Business Process Model) es simplemente un conjunto de tipos de asignaciones (Allocation Types). El rombo negro al final de la línea que representa la relación entre Process Model y Allocation Type implica que se trata de una relación de composición.

Según cómo se instancien las clases que forman parte del metamodelo, pueden obtenerse distintas familias de modelos. Por ejemplo, podemos modelar procesos en los cuales el tipo de valor relacionado depende tan sólo del tipo de infraestructura a la que se accede; o bien modelos en los que el que el tipo de valor depende de qué tipo de segmento de cliente accede a la oferta de infraestructuras. También se admiten otras variantes. De esta forma, el metamodelo puede dar soporte a la definición de un modelo de proceso de negocio en el cual la asignación de precios a las habitaciones de un hotel se realiza exclusivamente según el tipo de habitación que se pretende reservar, aunque también a modelos alternativos en los que la pertenencia del cliente a un programa de fidelización de la cadena de hoteles puede suponer una rebaja en el precio de la habitación con respecto a lo que se les cobra a los clientes no habituales.

En el ejemplo anterior, el precio se corresponde con el tipo de valor (Value Type) perteneciente al tipo de asignación (Allocation Type) que se oferta. El tipo de habitación es un tipo de infraestructura (Infrastructure Type) y el grupo de clientes que forman parte de una determinada categoría del programa de fidelización de la cadena hotelera es un tipo de segmento de clientes (Customer Segment Type). Aunque adoptar la industria hotelera como fuente de ejemplos es habitual en los textos que versan sobre asignación de infraestructuras, el Process Metamodel es válido para generar modelos de proceso de asignación de infraestructuras en otros sectores, como el sanitario, donde los pacientes con un determinado diagnóstico podrían formar parte de un mismo tipo de segmento de cliente (Customer Segment Type), los tipos de canal (Channel Types) podrían ser la ambulancia o el triaje de urgencias y los médicos de las distintas especialidades ser considerados como diferentes tipos de infraestructura (Infrastructure Types).

2.1.2 MODELO DE PROCESO DE NEGOCIO

Como se ha explicado en la sección precedente, un modelo de proceso de negocio es un instanciación directa del metamodelo descrito en la sección anterior. Para

exponerlo con claridad, nos servimos de un ejemplo de un modelo de reservas simplificado perteneciente a un establecimiento alquiler de automóviles.

Consideramos que existen dos tipos de segmento de clientes (Customer Segment Types), mayor de 25 años y menor de 25 años, a los que los clientes pertenecen de manera completa y disjunta según su edad. Es decir, todo cliente pertenece a uno de estos dos segmentos y sólo a uno. Se pueden realizar las reservas a través de un único tipo de canal (Channel Type), el sitio web del establecimiento. Se ofertan dos tipos de automóviles, o dicho de otra forma, dos tipos de infraestructura (Infrastructure Type), coche compacto y coche de alta gama. Existen tres tipos de valor (Value Types), Precio bajo, Precio medio y Precio alto; representan los precios por día de utilización de un automóvil, en orden ascendente. El proceso de negocio diseñado por el equipo de gestión del establecimiento de alquiler de vehículos contempla que, en primer lugar, sólo se ofrezcan los coches de alta gama a los mayores de 25 años. y que además entre el momento de tiempo genérico de inicio (T.Start) y un tiempo genérico cualquiera entre el tiempo genérico de inicio y el tiempo genérico de fin (T1), se haga a un precio medio; mientras que entre T1 y el tiempo genérico de fin (T.End) sea a un precio alto. Los coches compactos se ofrecen a todos los clientes, con un precio bajo entre T.Start y otro tiempo genérico cualquiera entre T.Start y T.End (T2) y a un precio medio desde T2 a T.End. El proceso de negocio descrito se representa en la siguiente tabla:

Customer Segment Type	Channel Type	Infrastructure Type	Value Type	Generic Start	Generic End
Menor de 25 años	Sitio Web	Coche compacto	Precio bajo	T.Start	T1
Menor de 25 años	Sitio Web	Coche compacto	Precio medio	T1	T.End
Mayor de 25 años	Sitio Web	Coche compacto	Precio bajo	T.Start	T1
Mayor de 25 años	Sitio Web	Coche compacto	Precio medio	T1	T.End
Mayor de 25 años	Sitio Web	Coche de alta gama	Precio medio	T.Start	T2
Mayor de 25 años	Sitio Web	Coche de alta gama	Precio alto	T2	T.End

Tabla 2-1. Ejemplo de un proceso de reserva de coches de alquiler

2.1.3 INSTANCIA DE PROCESO DE NEGOCIO

Se define una Instancia de Proceso de Negocio (Business Process Instance) a través de la configuración de los atributos pertenecientes a cada elemento del modelo de proceso de negocio y la adopción de valores por parte de los parámetros existentes. Si continuamos con el ejemplo del establecimiento de alquiler de coches, el concepto de Business Process Instance da cabida a diferentes formas de abordar un problema de optimización del sistema de reservas:

- Dada una distribución de llegadas para cada segmento de clientes, la cantidad de coches disponibles inicialmente de cada tipo, los distintos niveles de precios, y el momento T1 en el cual los coches compactos cambian de precio, se puede determinar cuál es el valor óptimo que debe tomar T2 para que el beneficio del establecimiento del alquiler de automóviles sea máximo.
- Dada una distribución de llegadas para cada segmento de clientes, la cantidad de coches disponibles inicialmente de cada tipo, los momentos T1 y T2 en los cuales los distintos tipos de coches cambian de precio, y los niveles de precio bajo y medio, se puede determinar cuál es el valor óptimo que debe tomar el nivel de precio alto para maximizar el beneficio del establecimiento de alquiler de automóviles.

El nivel de Process Instance, como ya se ha comentado, se corresponde con el tercer nivel jerárquico del modelo, cuyas dos primeras capas las integraban el metamodelo y el modelo de procesos. Tal y como se aprecia en la figura 2-2, el nivel de instancia de proceso incluye, entre otras, las siguientes clases: segmento de cliente (Customer Segment), canal (Channel), conjunto de infraestructura (Infrastructure Set), valor (Value) y tiempo (Time). El significado de las relaciones del tipo *Is Instance of* es el siguiente: las instancias de estas clases son instancias de las instancias de las clases del nivel superior, es decir, tipo de segmento de cliente (Customer Segment Type), tipo de canal (Channel Type), tipo de infraestructura (Infrastructure Type), tipo de valor (Value Type) y tiempo genérico (Generic Time), respectivamente. Cabe mostrar un ejemplo explicativo: si se dispone de la clase tipo de mueble y de la clase mueble, las instancias de la clase mueble serán instancias de las instancias de la clase tipo de mueble. La clase silla es instancia de la clase tipo de mueble, una instancia de la clase mueble puede ser una silla concreta y a su vez esa silla es instancia de la clase silla.

Las relaciones descritas son del tipo uno a muchos, es decir, que, por ejemplo, en una misma instancia de proceso puede existir más de un segmento de cliente, y además un mismo tipo de segmento de cliente puede instanciarse en varios segmentos de cliente que pertenecen a instancias de proceso distintas.

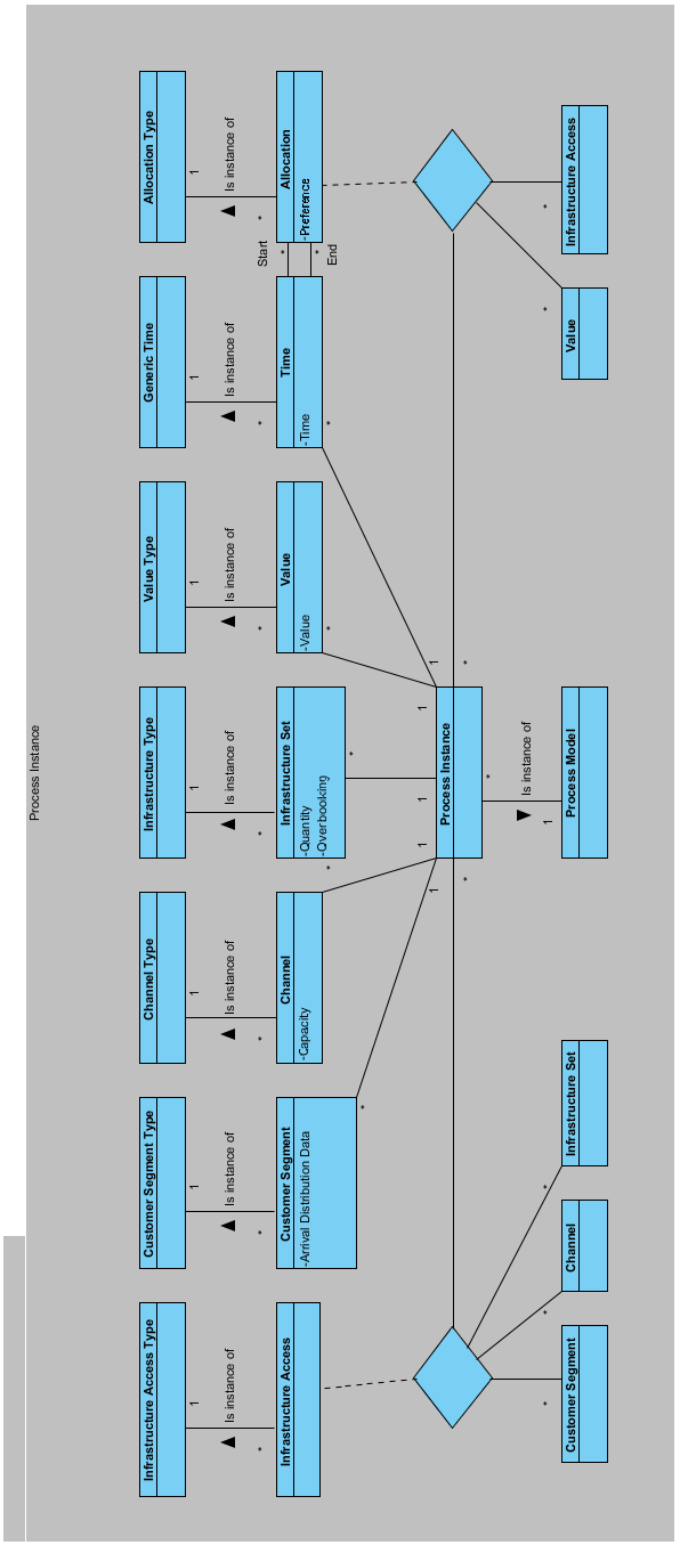


Figura 2-2. Diagrama de clases UML de Process Instance en el modelo de partida

El acceso de los clientes de un segmento, por medio de un canal, para reservar un conjunto de una infraestructura se conoce como acceso a una infraestructura (Infrastructure Access). Por coherencia, las restricciones al acceso que imperaban en el nivel superior, siguen vigentes para las instancias directas de cada una de las entidades. Por ejemplo, si se no existía la posibilidad de reservar algunos tipos de infraestructura (Infrastructure Types) a través de ciertos tipos de canal (Channel Types), las instancias de los conjuntos de infraestructura (Infrastructure Sets) que son instancia de las instancias de esos tipos de infraestructura (Infrastructure Types) no pueden reservarse a través de las instancias de los canales (Channels) que son instancia de las instancias de los tipos de canal (Channel Types) mencionados. No obstante, dado que cada segmento de cliente (Customer Segment), canal (Channel) y conjunto de infraestructura (Infrastructure Set) no son instancias únicas de sus correspondientes padres, existen varios accesos a infraestructuras (Infrastructure Access) idénticos, pero asociados a distintas instancias de proceso.

A cada instancia directa de la clase acceso a una infraestructura (Infrastructure Access) se le asigna un valor (Value). Esta asignación forma parte de la definición de una asignación (Allocation). Cada asignación está vigente durante un intervalo de tiempo cuyos extremos inferior y superior son respectivamente el comienzo (Start) y el final (End). Estos tiempos o bien toman un valor concreto en este nivel, o bien pueden ser una variable, tal y como se explicará en el siguiente apartado. Los valores concretos que adoptan los tiempos deben cumplir las relaciones de comparación definidas en genérico en el nivel superior, denominadas restricciones (Constraints).

2.2 DISEÑO DE LA HERRAMIENTA

El diseño de la herramienta debe estar alineado con el objetivo de su implementación, que es conseguir un software que permita, de forma flexible y cómoda, plantear problemas de asignación de infraestructuras en los que un cliente, perteneciente a un segmento, accede por un canal a la reserva de una infraestructura.

Para ello, se plantea un diseño formado por cuatro módulos, que cumplen, por sí solos, con el objetivo del proyecto. Pueden agruparse según los niveles jerárquicos del modelo conceptual a los cuales pertenecen.

- Niveles jerárquicos de metamodelo y modelo de proceso de negocio:
- Módulo de definición de Procesos de Negocio. Este primer módulo se construye sobre la base conceptual de los dos primeros niveles jerárquicos expuestos en la sección 2.1 del presente documento. Basado en los elementos del metamodelo de proceso, permite configurar un modelo mediante instancias directas de dichos elementos. Esto es, habilita la creación de tipos de segmentos de clientes, tipos de canal, tipos de infraestructuras, tipos de acceso a una infraestructura, tipos de valor, tiempos genéricos, restricciones temporales y tipos de asignaciones.

- Nivel jerárquico de instancia de proceso de negocio:

- Módulo de definición de Atributos. Junto a los siguientes dos módulos, forma parte de la creación de una instancia de proceso de negocio. El cometido de este módulo es el de dar soporte al usuario para especificar qué atributos se asocian con qué elementos de la instancia y cuál es su naturaleza. Es decir, el interés de este módulo reside en que permite detallar cuáles son los atributos asociados a los segmentos de clientes, a los canales, a los elementos de infraestructuras, etc., elegir qué tipo de información guardan y además clasificarlos como parámetros o como variables del problema.
- Módulo de definición de Estados. La inclusión de este módulo surge como respuesta a la necesidad de contar con una descripción de la dinámica de la instancia de proceso que permita definir una función objetivo de la forma más completa y realista posible. Si bien su existencia está supeditada al siguiente módulo, se explica antes porque se sigue el orden que mantienen los distintos módulos en la aplicación. Se basa en la definición de los estados en los que puede encontrarse un acceso o una asignación, de los cambios de estado que pueden producirse y de las características de los eventos que provocan dichos cambios de estado.
- Módulo de definición de la Función Objetivo. Es el módulo con el que culmina la configuración de una instancia de proceso de negocio. Se sirve de los dos módulos anteriores para articular una propuesta genérica y muy versátil de definición de una expresión cuya optimización se pretende. El interés de este módulo reside en que la maximización o minimización de la función conduciría a la resolución del problema cuyo planteamiento se aborda.

Por una cuestión de alcance del proyecto no se incluye un módulo de simulación, que ya pertenecería a un nivel jerárquico inferior, llamado nivel de ejecución de proceso de negocio, y que permitiría resolver el problema que la herramienta diseñada permite plantear actualmente.

2.3 HERRAMIENTAS DE DESARROLLO

Como ya se ha anunciado en el capítulo de Introducción, se presentan a continuación, con un apartado dedicado a cada una de ellas, las tres herramientas de desarrollo fundamentales que se han utilizado en la realización del proyecto: el lenguaje de modelado UML, el lenguaje de modelado de datos IDEF1X y el lenguaje de programación Visual Basic.

2.3.1 UML

Como herramienta de modelado para el proceso de diseño conceptual se utiliza

el Lenguaje de Modelado Unificado (UML, Unified Modeling Language), desarrollado por la organización OMG (Object Management Group, grupo para la gestión de objetos) con la intención inicial de aunar y estandarizar la profusión de lenguajes de modelado de orientación a objetos aparecidos a final de la década de los 1980s y principios de la década de los 1990s (Fowler, 2003). Según la descripción del OMG, UML ayuda a especificar, visualizar y documentar modelos de sistemas software, incluyendo su estructura y diseño, si bien UML puede emplearse para modelar procesos de negocio y sistemas no software. La elección de UML proviene de la confluencia de dos características: por un lado, la capacidad para el modelado de procesos de negocio ligada a su orientación a los sistemas; y por otro, el carácter de estándar unificador con el que nace el lenguaje, y que ha favorecido su creciente difusión y aceptación.

En concreto, se emplean fundamentalmente los Diagramas de Clases, que surgen como una evolución de los modelos entidad-relación. Su función es describir los distintos tipos de elementos que forman parte de un sistema, conocidos como entidades. Estas entidades guardan relaciones entre ellas, un aspecto que también se refleja en estos diagramas. Las entidades se representan por medido de cajas de forma rectangular y las relaciones como líneas entre dos de estas cajas. Existen dos tipos básicos de relaciones: de asociación, en la que las entidades participantes se encuentran en un mismo nivel, y de generalización, de tal forma que una de las entidades es un subtipo de la otra. Las relaciones de generalización se señalan a través de una línea precedida de un triángulo que apunta a la entidad de nivel superior. En las relaciones se puede especificar una cardinalidad, que es el grado de multiplicidad con el que una entidad participa en una relación; se especifican con el número correspondiente o con un asterisco cuando se quiere significar que la multiplicidad puede ser cualquiera. Adicionalmente se puede anotar junto a la entidad el rol con el que una entidad participa de una relación.

También aparecen en este documento los Diagramas de Secuencia de UML. Su función es mostrar cuáles son las interacciones entre los objetos que forman parte de un sistema y en qué orden se producen. Presentan, a modo de bandas paralelas verticales, los diferentes procesos u objetos que están activos de forma simultánea y, como flechas horizontales, los mensajes que se intercambian entre ellos. En nuestro caso, se utilizarán para conocer qué formularios de la aplicación se encuentran activos en cada momento, por medio de qué métodos se produce la comunicación entre ellos y en qué orden.

El tercer y último tipo de diagramas UML empleados en la elaboración del proyecto son los Diagramas de Estado. Su misión es describir el comportamiento de un sistema desde el punto de vista de los estados en los que se pueden encontrar los objetos que pertenecen al mismo, resaltando cuáles son los eventos que dan lugar a cada tipo de cambio de estado. El punto de partida de lectura del gráfico viene señalado por un círculo negro. Los estados aparecen como rectángulos de esquinas redondeadas, con el nombre del estado en su interior. Las flechas que unen los estados representan los pasos de un estado a otro. Junto a cada flecha se muestra el evento que provoca el cambio de estado. Finalmente, el cierre de la sucesión de estados se representa por medio de un círculo negro en el interior de una circunferencia concéntrica.

2.3.2 IDEF1X

Entre los requisitos del proyecto, expuestos en el capítulo introductorio junto al objetivo, se menciona la necesidad de emplear algún tipo de método para traducir el modelo conceptual a modelo de datos. IDEF1X es la herramienta que puede ejercer esta función.

IDEF1X es un lenguaje de modelado de datos desde un punto de vista lógico. Destaca como herramienta estándar, predecible y coherente de ordenar la información y facilitar su gestión (Bruce, 1991). Su utilidad como asistente en la creación de bases de datos resulta destacable. Los principales elementos que caracterizan los diagramas IDEF1X son los siguientes:

- Entidad: Representa un conjunto de instancias de datos similares y que pueden ser unívocamente identificadas de forma individual (Date, 2009). Gráficamente, aparecen en los diagramas en forma de rectángulos. Las esquinas son puntiagudas si la entidad es independiente, esto es, si no depende de otra para ser definida. Por el contrario, si la entidad es dependiente, es decir, si no es concebible sin relacionarla con otra u otras entidades, las esquinas del rectángulo son redondeadas.
- Atributo: Es una propiedad que contribuye a definir la naturaleza de una entidad. Lo razonable es que el nombre de los atributos dé una idea de su significado.
- Clave: Agrupa a una serie de atributos que identifican de forma unívoca una instancia de una entidad, es decir, un registro de una tabla. Los atributos que forman parte de la clave aparecen en la mitad superior de la entidad.
- Relación de conexión: Muestra cómo una entidad se relaciona con otra. Se da siempre entre dos entidades, y sólo dos. Se representa por medio de una línea que une ambas entidades. Si la relación es identificativa, esto es, si la clave del padre pasa a formar parte de la clave del hijo, la línea es continua. En caso de que la relación sea no identificativa, situación que se da cuando la clave del padre no pasa a formar parte de la clave del hijo sino que aparece entre el resto de los atributos en la mitad inferior de la entidad, la línea utilizada es discontinua (Rob y Colonel, 2006).
  Todo atributo heredado del padre aparece junto a la abreviatura FK (Foreign Key). Además, junto al hijo, al final de la línea, aparece un círculo negro, que indica la cardinalidad de la relación.
- Relación de categorización: Permite definir una clasificación que engloba a algunas de las entidades del modelo como pertenecientes a una entidad que se halla en un nivel superior de generalidad. Se representa con la ayuda de un círculo blanco, que tendrá una o dos líneas horizontales debajo dependiendo de si la clasificación en incompleta o completa, respectivamente. Junto a dicho círculo aparece el nombre del atributo que actúa como discriminador.

2.3.2 VISUAL BASIC

Tal y como se expresa en la sección del capítulo introductorio dedicada al objetivo del proyecto, la interfaz gráfica de usuario debe permitir la definición flexible, rápida e intuitiva de los modelos e instancias de proceso. La solución adoptada para hacer frente a tal necesidad es Visual Basic.

Se trata de un lenguaje de tercera generación guiado por eventos. Está orientado al desarrollo rápido de aplicaciones con interfaz gráfica de usuario (GUI). Su diseño facilita el aprendizaje rápido por parte de programadores inexpertos, lo cual no quiere decir que no sea también un entorno de programación muy potente y adecuado para la implementación de proyectos más complejos. En lo que respecta a la configuración de la apariencia visual del programa, el programador puede seleccionar con el ratón los objetos que van a formar parte de cada formulario o ventana de entre una serie de objetos preconfigurados, dimensionarlos y situarlos de forma muy intuitiva, así como modificar muchas de sus propiedades sin escribir una sola línea de código. Tanto su versatilidad como su facilidad para el aprendizaje son características de Visual Basic que determinaron su elección como lenguaje para abordar la implementación de este proyecto. Su uso extendido y la gran cantidad de documentación y material de ayuda disponible son otros aspectos a destacar a su favor.

A cada formulario y los objetos que pertenecen al mismo (denominados controles), se asocia un conjunto de eventos predefinidos por Visual Basic. Ejemplos de eventos son hacer click sobre un botón, modificar el tamaño de un control o cerrar un formulario. No todos los eventos se deben a la acción directa del usuario. Los eventos pueden desencadenarse por mensajes del sistema operativo, de la propia aplicación o de otras aplicaciones (Gaddis e Irvine, 2010). A su vez, cada uno de estos eventos está ligado a unas líneas de código. Se trata de instrucciones que se ejecutan durante el uso de la aplicación si sucede el evento correspondiente. Así mismo, estas instrucciones pueden determinar la ocurrencia de un evento. Es posible que existan eventos sin código asociado. De hecho, será la situación de la mayor parte de los eventos.

Dado que los eventos, en general, no se encuentran relacionados por ningún tipo de secuencia, el programador deberá tener en cuenta esta circunstancia y poner en práctica una programación inteligente y a prueba de errores del usuario. Para ello, existen atributos asociados a los controles que permiten desactivar eventos asociados a los mismos hasta que no se desencadenen eventos previos. Ejemplos de estos atributos especialmente útiles son Visible (visible) o Enabled (activado), que cuando adquiren el valor False (falso), no permiten al usuario hacer click sobre el control al cual están asociados.

3

MÓDULO DE DEFINICIÓN DE PROCESOS DE NEGOCIO

En el presente capítulo se exponen, en primer lugar, las diferencias entre el modelo de partida de procesos de negocio y el diseño conceptual del módulo de procesos de negocio de la herramienta informática. La segunda parte del capítulo corresponde al modelo de datos. Se analiza el diseño de la base de datos que procura soporte a la aplicación, incluyendo un diagrama explicativo de la misma. El tercer apartado versa sobre el diseño de la aplicación, incluyendo la justificación de su estructura y el diagrama de estados correspondiente. Por último, se expone cómo se ha desarrollado la implementación del módulo del software correspondiente a la configuración de un proceso de negocio.

3.1 DISEÑO CONCEPTUAL

El diseño conceptual del módulo de definición de procesos de negocio está inspirado en el modelo de partida que se explicó en el segundo capítulo de este documento. La única modificación significativa que se ha introducido se refiere al concepto de entidad del modelo (Model Entity).

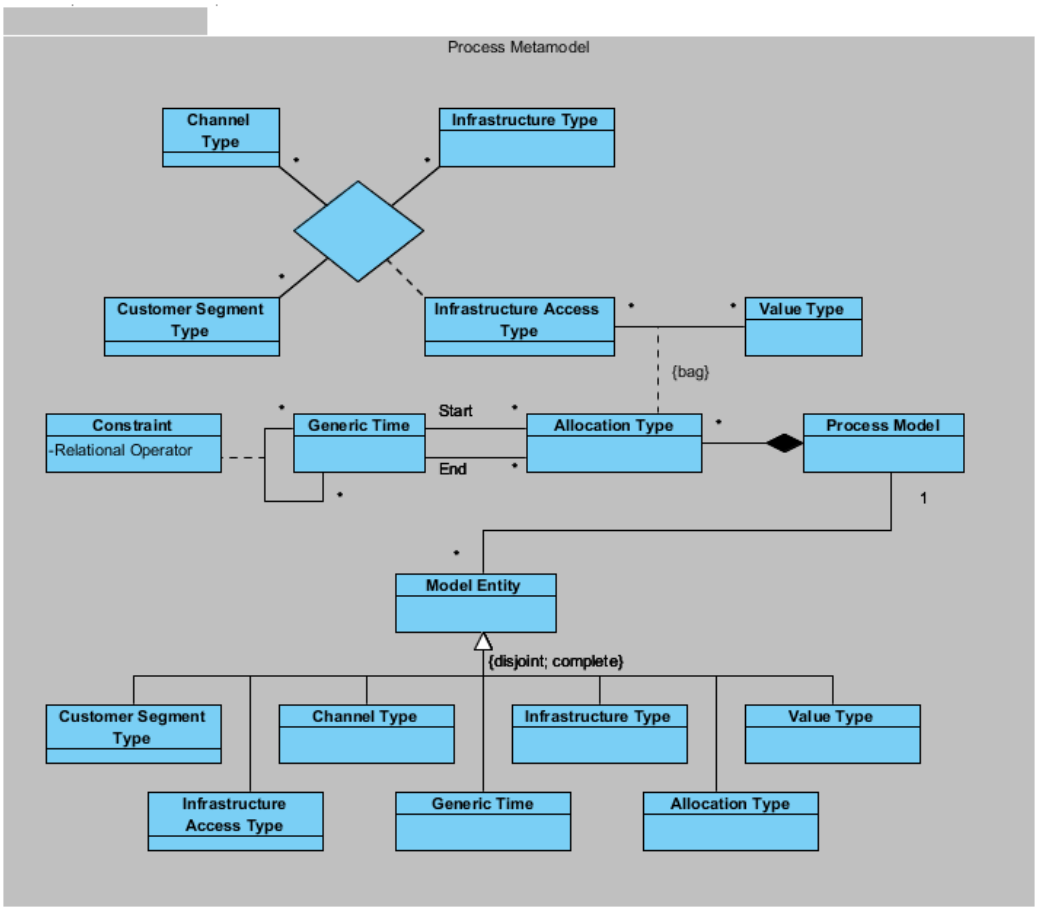


Figura 3-1. Diagrama de clases UML del Process Metamodel en el diseño conceptual del primer módulo de la herramienta informática

En cada modelo de proceso de negocio (Business Process Model) existen varias entidades del modelo (Model Entities), que son los elementos básicos que se relacionan para dar lugar al modelo y que cuyo significado hemos explicado uno a uno con anterioridad en la introducción teórica del modelo de partida. Las entidades del modelo (Model Entities) pueden clasificarse de forma disjunta y completa como tipo de segmento de cliente (Customer Segment Type), tipo de canal (Channel Type), tipo de infraestructura (Infrastructure Type), tipo de valor (Value Type), tipo de acceso a una infraestructura (Infrastructure Access Type), tiempo genérico (Generic Time) y tipo de asignación (Allocation Type). Es decir, toda Model Entity pertenece forzosamente a uno de esos tipos y sólo a uno.

La modificación con respecto al modelo de partida se ha introducido debido a la necesidad de vincular la existencia de cada elemento de los anteriormente enumerados a un modelo de proceso concreto. Puesto que la herramienta está orientada al modelo de proceso y su existencia es requisito para definir cualquier tipo de elemento, tal vinculación resulta imprescindible. No tiene cabida la existencia de cualquiera de estos elementos fuera de un modelo de proceso.

3.2 MODELO DE DATOS

En esta sección se expone el modelo de datos de este módulo. Para ello se detalla su diseño lógico, con la asistencia del diagrama IDEF1X del segmento de la base de datos correspondiente a la definición del modelo de proceso (Process Model), elaborado con ERWin, y que se muestra en la figura 3-2.

La primera decisión importante en el diseño de la parte de la base de datos relacionada con la definición del Process Model, es decidir qué tablas son necesarias. Se considera que a cada una de las entidades que aparecen en el diagrama de clases Process Metamodel, es necesario asignarle una tabla propia en la base de datos. Así pues, las tablas necesarias son: ProcessModel, ModelEntity, CustomerSegmentType, ChannelType, InfrastructureType, InfrastructureAccessType, ValueType, GenericTime, TableConstraint y AllocationType, tal y como puede apreciarse en la figura anterior.

El hecho de que la denominación de la tabla de la base de datos asociada a la entidad Constraint sea TableConstraint se debe a que “Constraint” es una palabra clave en el lenguaje SQL (lenguaje de consulta de la base de datos que utilizaremos), por lo que su uso como nombre de una tabla da lugar a errores.

La tabla en la cual se anota el primer registro cuando se guarda un modelo de proceso de negocio es ProcessModel. Se trata de la única entidad independiente de todas las que aparecen en el diagrama de la figura anterior. La clave primaria de esta tabla es numérica y propia. Se denomina ID_ProcessModel. Además la tabla cuenta con un campo adicional, Name_, un atributo que se corresponde con el nombre que el usuario decide darle al Process Model. El guión bajo del final es necesario porque “Name” es una palabra clave del lenguaje SQL, al igual que sucedía con “Constraint”.

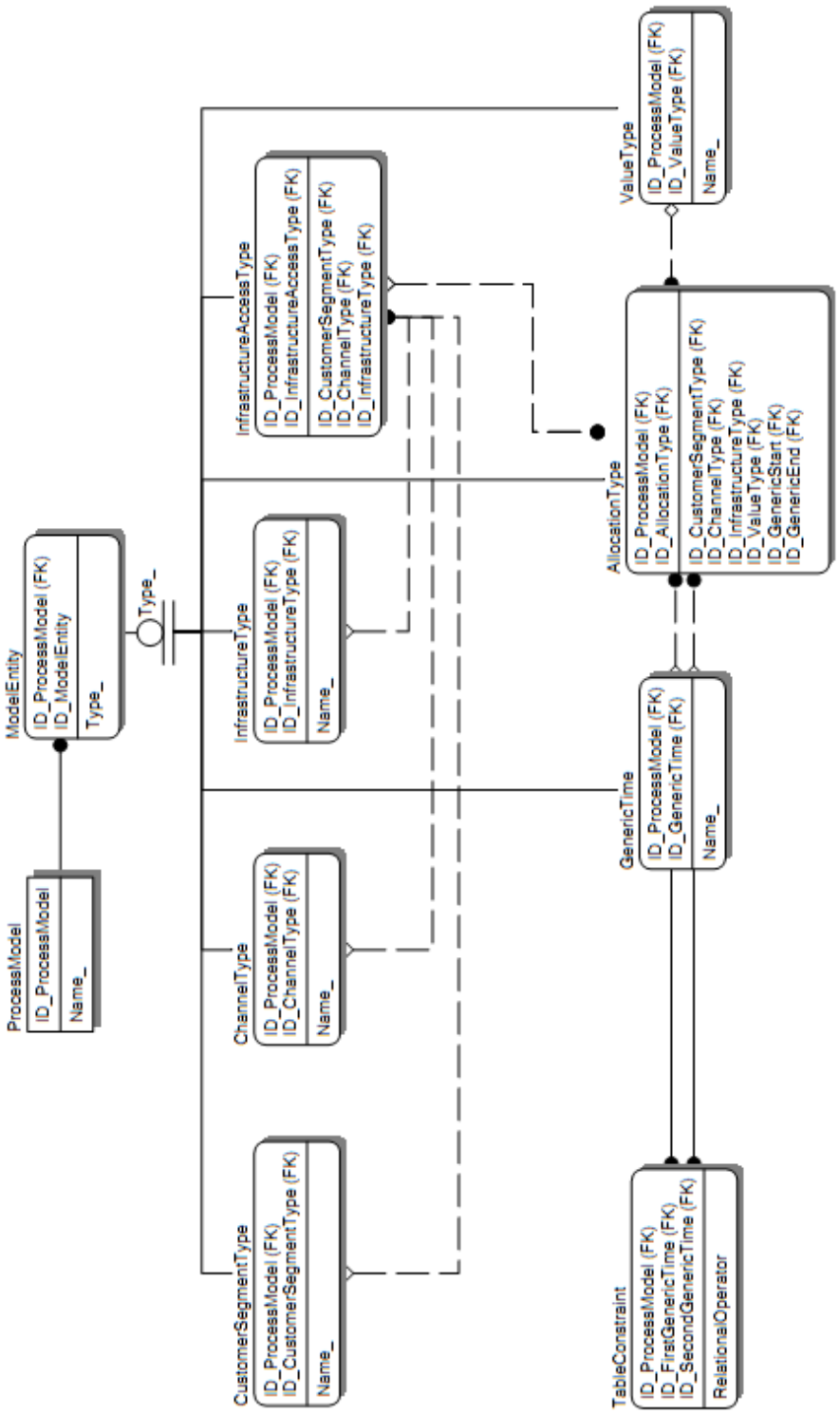


Figura 3-2. Diagrama IDEF1X del segmento de la base de datos correspondiente al Process Model

Las Model Entities son entidades básicas presentes en el proceso de modelo de negocio. Pueden ser de distintos tipos de forma exclusiva y excluyente: Customer Segment Type, Channel Type, Infrastructure Type, Value Type, Generic Time, Infrastructure Access Type o Allocation Type. Cada Model Entity se asocia a un Process Model concreto. La clave de la tabla ModelEntity está formada por una clave propia, ID_ModelEntity, y una clave ajena, ID_ProcessModel. Además, el atributo Type_ describe el tipo de Model Entity al cual pertenece el registro, es decir, actúa como discriminador en la clasificación. En consecuencia, en el diagrama IDEF1X aparece junto al símbolo de la relación de categorización, que como hemos indicado anteriormente, es un circulo blanco, con una o dos líneas debajo según la completitud de la clasificación. De esta forma, cuando es necesario cargar un modelo de proceso de negocio desde la base de datos, el proceso es mucho más intuitivo y sencillo.

En principio pudiera pensarse que no es necesario implementar las tablas CustomerSegmentType, ChannelType, InfrastructureType ni ValueType, ya que cuando se requiriese la información correspondiente a las entidades asociadas a dichas tablas, lo cual sucede cada vez que el usuario carga un modelo o define una instancia del mismo, ésta podría reconstruirse fácilmente a partir de los registros de la tabla ModelEntity. Sin embargo, con vistas a no dificultar la expansión del metamodelado, se ha decidido dotar a la base de datos con las tablas mencionadas con anterioridad. La razón es que, en un momento dado, podría resultar conveniente asignar a alguna o a varias de estas entidades atributos propios o que no tuvieran sentido para las demás. Si bien es cierto que sería posible añadir tantos atributos extra en la tabla ModelEntity como fuera necesario y permitir en ellos valores nulos, lográndose el mismo efecto práctico, no sería la solución más elegante desde el punto de vista conceptual.

Los diferentes CustomerSegmentType, ChannelType, InfrastructureType y ValueType están identificados por medio de una clave primaria formada por una clave numérica ajena heredada (ID_ModelEntity, que pasa a denominarse ID_CustomerSegmentType, ID_ChannelType, ID_InfrastructureType e ID_ValueType, según el caso), y la clave ajena del Process Model correspondiente, heredada a través de la misma relación. El nombre no es más que un mero atributo, cuyo identificador es Name_.

Cada registro de la tabla InfrastructureAccessType cuenta con una clave primaria formada por una clave ajena, ID_InfrastructureAccessType, que coincide con su clave en la tabla ModelEntity (ID_ModelEntity) combinada con una clave ajena correspondiente a su Process Model (ID_ProcessModel), heredada a través de ModelEntity. Cada registro de esta tabla se corresponde con una combinación de un CustomerSegmentType, un ChannelType y un InfrastructureType en el contexto de un determinado ProcessModel. Es decir, una instancia de la entidad InfrastructureAccessType, que hereda como claves no primarias las claves ajenas de los CustomerSegmentType, ChannelType e InfrastructureType correspondientes.

Un registro de la tabla AllocationType se corresponde con una combinación de un CustomerSegmentType, un ChannelType y un InfrastructureType, ligada a su vez a

un determinado ValueType, que estará vigente entre dos puntos temporales genéricos o GenericTimes (GenericStart y GenericEnd) de un determinado ProcessModel. La clave primaria de un registro de la tabla AllocationType es la clave numérica ajena ID_AllocationType (que coincide con su clave ID_ModelEntity en la tabla ModelEntity) combinada con la clave del ProcessModel al cual está asociado ese AllocationType, heredada a través de la relación de categorización con la entidad ModelEntity. Además, cada registro cuenta como claves no primarias con las claves ajenas de los CustomerSegmentType, ChannelType, InfrastructureType, ValueType, GenericStart y GenericEnd correspondientes.

Los registros de la tabla GenericTime se corresponden con tiempos genéricos que actúan como momentos de inicio y de fin de vigencia de los distintos AllocationType. Su clave está formada por la clave ajena ID_GenericTime, de tipo numérico y que coincide con la clave ID_ModelEntity correspondiente en la tabla ModelEntity, y la clave ajena ID_ProcessModel del modelo de proceso de negocio al cual pertenece ese tiempo genérico, heredada a través de su relación de categorización con ModelEntity. Además, cada Generic Time cuenta con un atributo Name_, que permite guardar el nombre del mismo. En todo Process Model completamente definido existen, como mínimo, dos Generic Time, denominados T.Start y T.End, que se asimilan con los tiempos genéricos de inicio y fin de actividad del proceso de negocio que se pretende modelar. Adicionalmente, el usuario puede añadir GenericTimes extra, cuya denominación será T1, T2, y así sucesivamente.

Un registro de la tabla TableConstraint representa la restricción que puede establecerse entre dos instancias de la entidad GenericTime a través de un operador relacional (en nuestro caso las posibilidades son "≤" y "="). La tabla TableConstraint carece de clave propia, es decir, toda su clave es heredada. La misma está formada por tres claves ajenas, pertenecientes a cada uno de los GenericTime que aparecen en la expresión analítica de la restricción y al ProcessModel asociado a ambos. En consecuencia, para que cada registro pueda ser unívocamente identificado por medio de este sistema, la aplicación no le permitirá al usuario definir una restricción entre dos registros de GenericTime que aparezcan juntos en una restricción ya precisada, aunque sea con distinto operador relacional, el cual tiene carácter de atributo.

3.3 DISEÑO DE LA APLICACIÓN

En esta sección se expone el proceso de diseño del módulo la aplicación que da soporte a la definición de un modelo de proceso. Se desarrolla cuál es la estructura y se muestra un diagrama de secuencia de UML con la sucesión de los formularios, en la figura 3-3.

La aplicación debe contar con una pantalla inicial. Se pretende que esta ventana cumpla con dos funciones. Por un lado, facilitar al usuario la adquisición de una serie de nociones básicas sobre el cometido del software y el modo de utilizarlo. Por otra parte, debe servir como punto de partida para que el usuario pueda proceder con facilidad al

uso de la aplicación, permitiéndole, como mínimo y en este estadio del desarrollo, crear un nuevo modelo, cargar un modelo ya existente y borrar un modelo guardado anteriormente. Se considera que una forma adecuada de conseguirlo es con un menú desplegable en el cual el usuario pueda elegir entre una de las opciones anteriores.

Una vez el usuario decide que quiere crear un nuevo modelo de proceso, el programa debe solicitarle un nombre para el modelo y que además sea único, para que no haya posibilidad de confusión con otros modelos. Se descarta la opción de que el usuario pueda introducir el nombre de un modelo ya existente y que el programa le dé la posibilidad de sobrescribir, ya que no se desea que se produzcan pérdidas involuntarias de información valiosa o cuya introducción ha sido muy laboriosa.

La creación de un nuevo modelo impone que en el siguiente formulario se haga referencia a las cuatro primeras piezas básicas del modelo conceptual, es decir, a las cuatro Model Entities que forman la base y son: Customer Segment Type, Channel Type, Infrastructure Type y Value Type. Las primeras tres son imprescindibles para que en el siguiente formulario se pueda seguir profundizando en el modelo y se puedan definir los Infrastructure Access Types. El Value Type se incluye en este primer formulario de definición de un Process Model porque es una entidad de una naturaleza muy similar a las tres primeras. Sólo se solicita al usuario su nombre, no como ocurre con los Generic Times, que van ligados a una serie de restricciones que deben definirse. Además, no es una combinación de otras Model Entities, como les ocurre a Infrastructure Access Type y a Allocation Type.

Así pues, se decide el usuario tenga a su disposición cuatro listas, cada una de ellas correspondiente a cada una de las siguiente cuatro entidades: Customer Segment Type, Channel Type, Infrastructure Type y Value Type. Los elementos que forman parte de estas listas deben ser modificables por parte del usuario, por lo que por medio de tres botones colocados junto a cada lista se le permite añadir una nueva entidad a la lista, editar su nombre o borrarla del modelo. Además, el formulario debe contar con un botón que permita al usuario pasar al siguiente formulario.

El segundo formulario de definición de un modelo de proceso debe permitir al usuario elegir cuáles son los Infrastructure Access Types que permite en su modelo. Debe ser como mínimo uno. Para ello, se pueden mostrar en una lista todas las posibles combinaciones de los Customer Segment Types, Channel Types e Infrastructure Types que ha definido en el formulario anterior y que al usuario se le permita seleccionar cuáles son las combinaciones elegidas y que pasarán a ser los tipos de acceso a una infraestructura. El formulario contará con cuatro botones. Dos que permitan seleccionar y deseleccionar todas las combinaciones, para mejorar la usabilidad del software, y otros dos botones para pasar bien al formulario anterior o bien al siguiente. Ofrecer al usuario la posibilidad de volver al primer formulario de definición de un modelo de proceso se debe a que puede que en un momento dado quiera añadir o eliminar alguna de las entidades del modelo guardadas y no lo haya detectado con anterioridad.

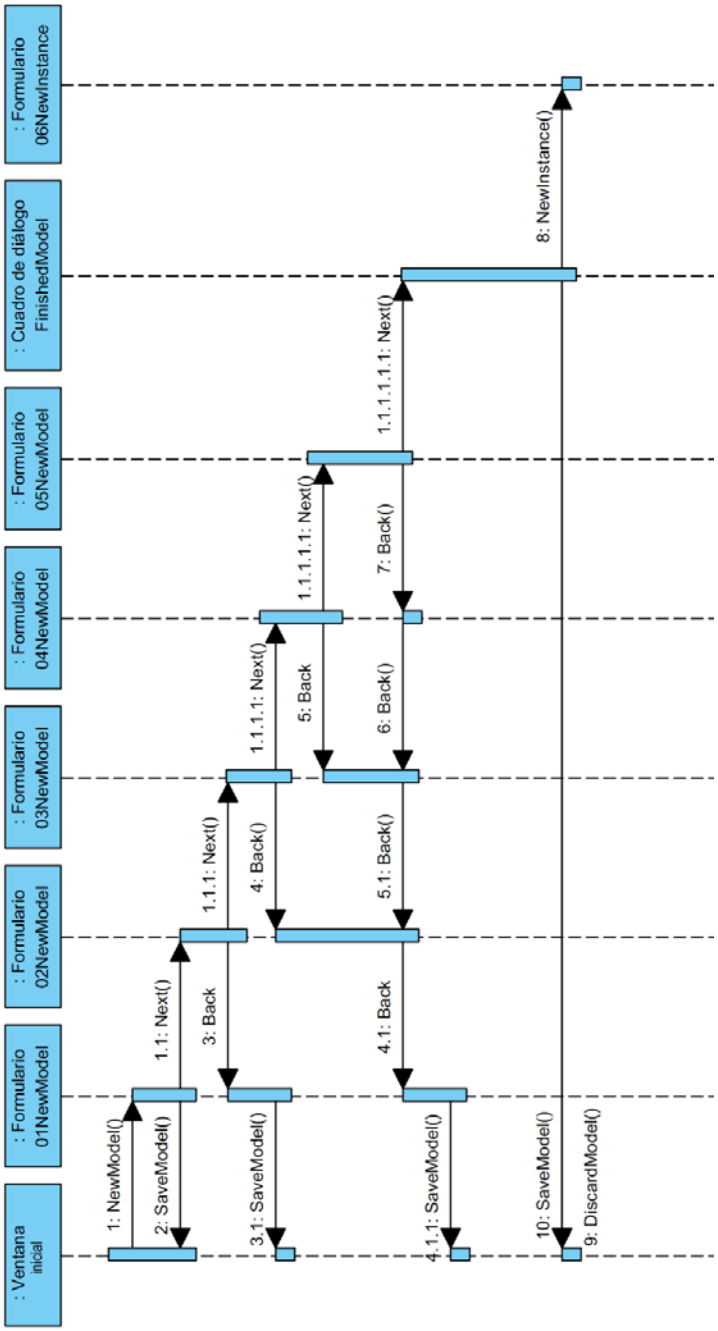


Figura 3-3. Diagrama de secuencia en UML del módulo de la aplicación de definición de un Process Instance

El tercer formulario tan sólo debe mostrar los Infrastructure Access Types en una lista y sin posibilidad de ser modificados, pero ya sin estar listados junto a todas las combinaciones de Customer Segment Types, Channel Types e Infrastructure Types obviadas por el usuario en el segundo formulario. Para que el usuario pueda corregir su error si detecta alguno, se debe habilitar un botón que le permite volver al formulario anterior. Así mismo, otro botón le permite acceder al siguiente formulario.

Puesto que ya disponemos de los Infrastructure Access Types y los Value Types, tan sólo nos falta un tipo de entidades del modelo para poder abordar la definición de los Allocation Types: se trata de los Generic Times. El cuarto formulario se destina a la definición de los mismos y de las Constraints que los ligan. Por un lado, el formulario debe contar con una lista en la que aparezcan como mínimo dos Generic Times: T.Start y T.End. Además, el usuario podrá añadir Generic Times adicionales con la ayuda de un botón, pero sin poder nombrarlos, algo que hará el software de manera automática. También podrá eliminar los Generic Times que ha añadido, uno a uno, con la asistencia de otro botón.

En el mismo formulario pero en otra parte del mismo, una lista debe mostrar cuáles son las restricciones que ligan a los Generic Times que aparecen en la otra lista. Como mínimo estas restricciones deben indicar que T.Start es menor o igual que el resto de los Generic Times y que T.End es mayor o igual que el resto de los Generic Times. A priori no tienen por qué existir restricciones que liguen entre sí los Generic Times que ha introducido el usuario, pero se le debe permitir añadir restricciones adicionales por medio de un botón y que además no contradigan a las restricciones ya definidas. Así mismo, por medio de otro botón, el usuario debe poder eliminar las restricciones que ha añadido. Finalmente, dos botones le deben permitir, respectivamente, pasar al formulario anterior o al siguiente.

El siguiente formulario, quinto y final de este módulo, da la oportunidad al usuario de definir los Allocation Types que van a componer su modelo de proceso. Se han barajado dos posibles diseños de este formulario.

El primero se basa en una lista de posibles Allocation Types, cada uno de ellos derivado de un Infrastructure Access, en los que son editables el Value Type, el Generic Start y el Generic End; y además se permite duplicar elementos en esa lista (y borrarlos posteriormente si se desea), por si el usuario desea definir dos o más Allocation Types que derivan de un mismo Infrastructure Access Type.

El diseño alternativo se basa en dos listas. En la primera se muestran los Infrastructure Access Type que el usuario definió en el segundo formulario. Junto a la misma existe un botón, de tal forma que si se pulsa tras haber seleccionado uno de los Infrastructure Access Type, aparece un cuadro de diálogo que da la posibilidad de elegir un Value Type, un Generic Start (Comienzo Genérico) y un Generic End (Final Genérico) y añadir un nuevo Allocation Type a la segunda lista, el cual cuenta con el Customer Segment Type, Channel Type e Infrastructure Type del Infrastructure Access Type elegido en primera instancia y el Value Type, Generic Start y Generic End seleccionados en el cuadro de diálogo. Además, dos botones bajo la segunda lista permiten editar o eliminar los Allocation Types que el usuario ha incluido en la misma.

Finalmente se optó por el segundo diseño, al resultar más amigable y más sencillo de entender para un usuario poco habituado a la utilización del programa. En ambos diseños se contempla en cualquier caso la existencia de un botón que permite volver al formulario anterior y otro que permite finalizar la definición del modelo de proceso. En caso de que opte por esta última posibilidad, un cuadro de diálogo permite

al usuario bien guardar el modelo y definir una instancia del mismo, bien simplemente guardar el modelo, o bien descartarlo y borrarlo.

3.4 IMPLEMENTACIÓN DE LA APLICACIÓN

En esta sección se describe la implementación de la aplicación cuyo diseño se ha desarrollado anteriormente. Para ello, se muestra la aplicación ya implementada, denominada RM Modeler: fundamentalmente la apariencia de los formularios que la forman y las funcionalidades asociadas a cada uno de ellos.

3.4.1 RM MODELER

El software permite al usuario definir un modelo de proceso de negocio (que se denomina Process Model), hacer lo propio con una instancia del modelo de proceso de negocio (Process Instance), incluyendo su parametrización, configuración de estados y establecimiento de una función objetivo. Se ha decidido que el proceso de definición del modelo y la instancia sea guiado por el software, minimizando la posibilidad de que se produzcan errores debidos al usuario y facilitando el aprendizaje de su uso. La interactuación con el programa se realiza por medio de ventanas, que tienen los controles genéricos y la apariencia de cualquier ventana de Windows.

A continuación, se expondrá la utilidad de cada una de las ventanas que componen el primer módulo del programa, el correspondiente a la definición de un proceso de negocio, además de mostrar su apariencia, describir sus componentes y facilitar las instrucciones oportunas para su uso.

3.4.1.1 VENTANA INICIAL

Se trata de la ventana de inicio, la primera que aparece cuando se ejecuta el programa RM Modeler. Su función es la de familiarizar al usuario con las instrucciones de uso de la aplicación, que se detallan en un texto que ocupa la parte central de la ventana (figura 3-4).

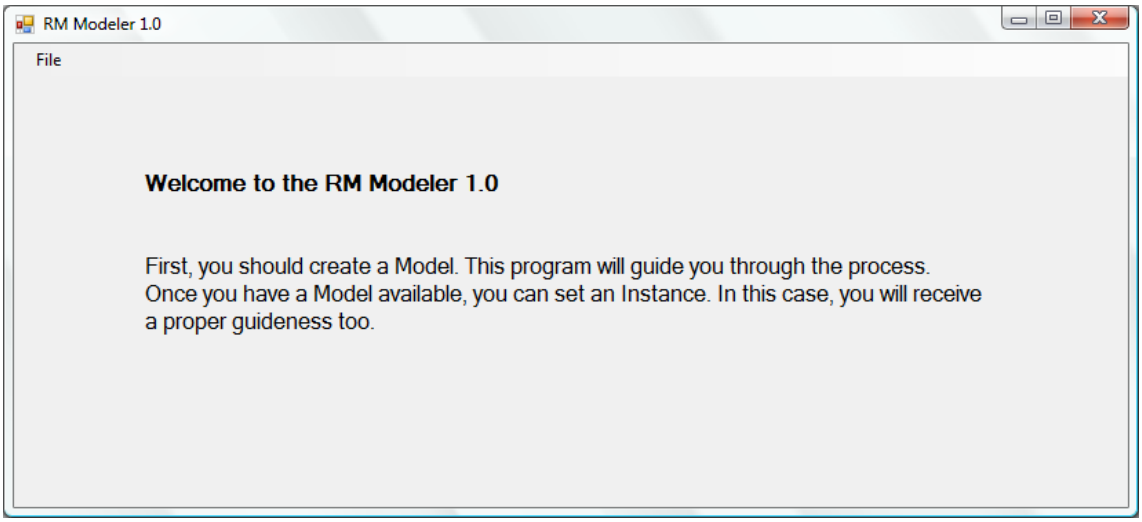


Figura 3-4. Ventana inicial del RM Modeler 1.0

Esta instrucciones son básicas y de carácter general. Indican al usuario cuáles son los pasos a seguir para hacer un uso efectivo del programa:

- Primero, debería crearse un modelo de proceso de negocio. En caso de que el usuario haya utilizado el software con anterioridad y haya definido ya un modelo de proceso de negocio (completamente o no), podría abrirlo y realizar los cambios que considerase oportunos.
- Una vez se cuenta con un modelo disponible, se puede definir una instancia del mismo.
- Por último, el software permite simular la ejecución de un proceso.

En esta pantalla, el usuario tan sólo puede interaccionar con el menú File.

Utilizar el menú File es la única alternativa que tiene el usuario de interaccionar con el programa partiendo de la ventana inicial. El usuario puede elegir entre cuatro opciones:

- New: Da la posibilidad de crear bien un nuevo Process Model o bien un nuevo Process Instance.
- Open: Permite al usuario abrir los Process Models y Process Instances creados con anterioridad.
- Delete: Sirve para borrar un Process Model, Process Instance o incluso toda la información almacenada en la base de datos.
- Exit: Para cerrar el programa.

Una vez el usuario, utilizando el menú File descrito en el apartado anterior, decide crear un nuevo Process Model, el programa le solicita el nombre del mismo por medio de un cuadro de diálogo. Si el usuario no inserta ningún nombre, o el que inserta comienza con un espacio, el programa le advierte que el nombre no es válido y la creación del Process Model se suspende. Además, cabe mencionar que este nombre no

podrá coincidir con el de ninguno de los Process Model ya almacenados en la base de datos. En caso de que el usuario viole esta condición, recibe un mensaje de aviso, explicándole lo sucedido y se le dirige a la ventana inicial (figura 3-5).

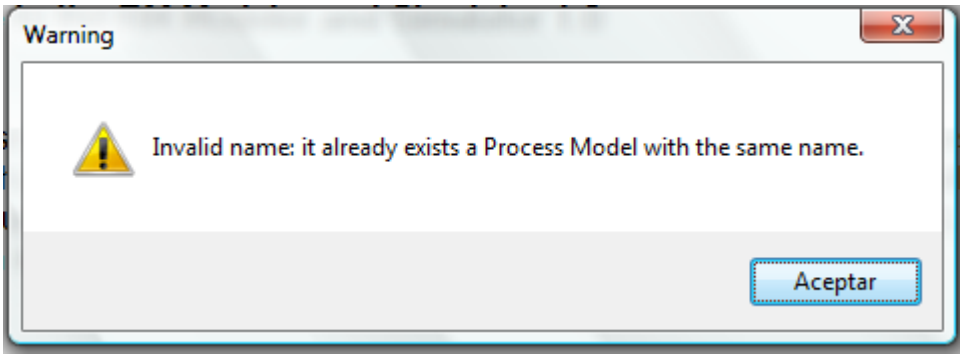


Figura 3-5. Mensaje que advierte que el nombre del nuevo Process Model no es válido porque ya está reservado

En caso de que el nombre introducido cumpla con las tres condiciones ya expuestas (más de un carácter, primer carácter distinto de espacio y nombre no coincidente con el de alguno de los nombres de los modelos de proceso de negocio almacenados en la base de datos), el programa muestra al usuario la primera ventana de definición del Process Model.

3.4.1.2 FORMULARIO DE DEFINICIÓN DE CUSTOMER SEGMENT TYPES, CHANNEL TYPES, INFRASTRUCTURE TYPES Y VALUE TYPES DE UN PROCESS MODEL

En la parte superior izquierda de dicha ventana se muestra el nombre del modelo elegido por el usuario (figura 3-6). Así mismo, en este formulario, se pueden introducir tantos Customer Segment Type, Channel Type, Infrastructure Type y Value Type como se desee. Para ello, se pulsa el botón Add existente junto al Model Entity que se quiere añadir. A continuación, un cuadro de diálogo solicita un nombre, que debe cumplir requisitos análogos a los del nombre del Process Model:

- Debe tener más de un carácter.
- El primer carácter no puede ser un espacio.
- No puede coincidir con el nombre de una entidad del mismo tipo ya definida en ese mismo Process Model. Por ejemplo, en dos Process Model distintos podemos disponer de dos Customer Segment Types con el mismo nombre, pero no en un mismo Process Model.

Si el usuario lo considera oportuno, puede editar el nombre de alguno de los Customer Segment Type, Channel Type, Infrastructure Type y Value Type añadidos con anterioridad. Para ello, debe pulsar sobre el botón Edit correspondiente, habiendo clickado previamente sobre la entidad que será objeto de la edición. Un cuadro de diálogo es el encargado de recoger el nuevo nombre. Éste debe cumplir con las condiciones anteriores o el cambio no se hará efectivo.

La tercera funcionalidad de este formulario es la que permite eliminar alguna Model Entity de las que se han creado previamente. Para ello, el usuario debe clickar sobre la entidad que desea borrar y pinchar sobre el botón Delete que se halla junto a la lista en la que aparece la entidad.

Esta ventana, al igual que la inicial, también dispone de un menú File, aunque en este caso con unas funcionalidades más reducidas. Permite guardar el Process Model en su estado actual de definición, utilizando la opción Save Process Model. En ese caso, se cierra la ventana actual, se almacena la información y el usuario es dirigido a la ventana inicial. Cabe mencionar que todos los cambios provocados por las acciones de los botones Add, Edit y Delete sobre los Customer Segment Types, Channel Types, Infrastructure Type y Value Types actualizan automáticamente en la base de datos. Además estas acciones tienen efectos sobre algunos formularios posteriores y tablas de la base de datos ligados a estos. Por ejemplo, eliminar un Channel Type, elimina también todos los Infrastructure Acces Types y Allocation Types en los que aparece.

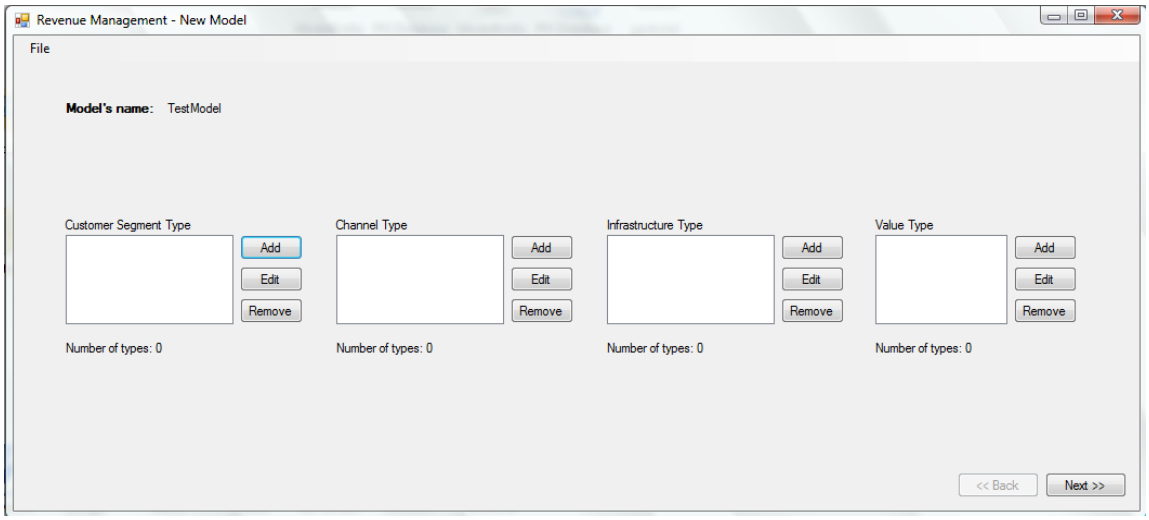


Figura 3-6. Primer formulario de definición de un Process Model, en blanco

También es posible clickar en Exit, con lo que se cierra el programa. El menú de esta pantalla es idéntico a los menús de todos los formularios de definición del Process Model, por lo que su composición y funciones no se abordarán en los demás apartados.

Por último, en la parte inferior derecha de la pantalla se dispone de un botón denominado Next, que ejecuta un código que, tras comprobar que existe al menos una Model Entity de cada uno de los cuatro tipos, permite el acceso al segundo formulario de definición de un Process Model. En la figura 3-6 puede apreciarse este botón y todos los elementos que se han descrito en este apartado. A modo de demostración se completa un Process Model de prueba para mostrar la apariencia que tendrá la pantalla una vez el usuario hay añadido las Model Entities que considere oportunas. El resultado se muestra en la figura 3-7.

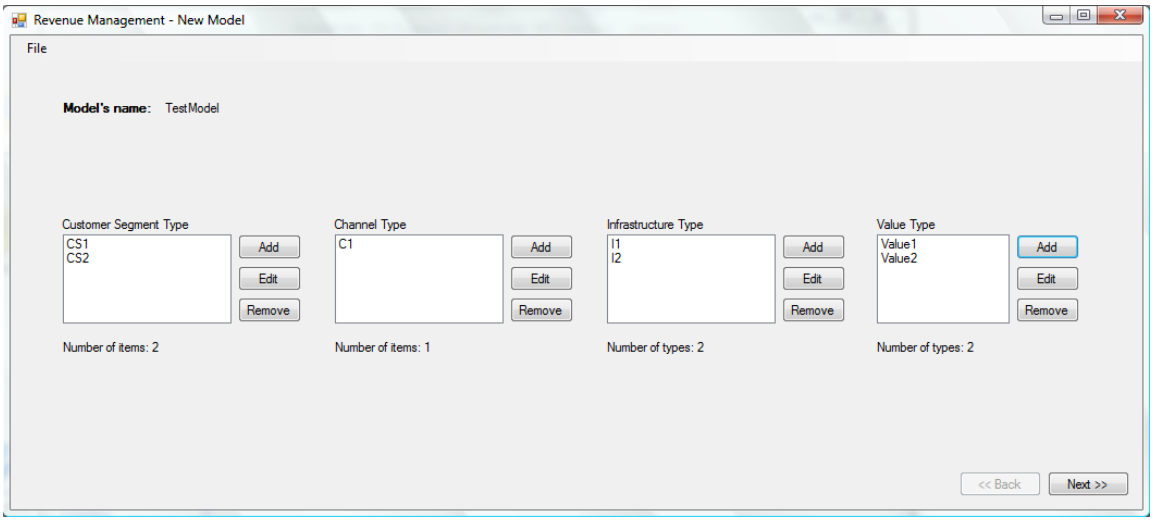


Figura 3-7. Primer formulario de definición de un Process Model, completada

3.4.1.3 FORMULARIOS DE ELECCIÓN DE LOS INFRASTRUCTURE ACCESS TYPES DE UN PROCESS MODEL

Los formularios número dos y tres de definición de un Process Model dan soporte al proceso de selección de los Infrastructure Access Type que el usuario desea habilitar en su modelo.

En el primero de ellos, que aparecen en la figura 3-8, se listan todas las posibles combinaciones de Customer Segment Types, Channel Types e Infrastructure Types que se han incluido en el modelo de proceso de negocio en el formulario anterior. Además, junto a cada combinación existe una casilla que permite marcar a la misma como seleccionada o no. Por defecto, la primera vez que se accede a esta página en un modelo de nueva creación, ninguno de los posibles candidatos a ser Infrastructure Access Type aparece como elegido. En el caso de que el modelo se haya definido anteriormente y se esté accediendo a una versión guardada del mismo, aparecerán como marcadas las casillas que lo estuviesen cuando se guardó el mismo, ya que al acceder a este formulario se realiza una carga desde la base de datos. Con el fin de mejorar la

usabilidad, en la parte inferior de la ventana están disponibles dos botones que permiten seleccionar (“Check all”) o deseleccionar (“Uncheck all”) todas las combinaciones.

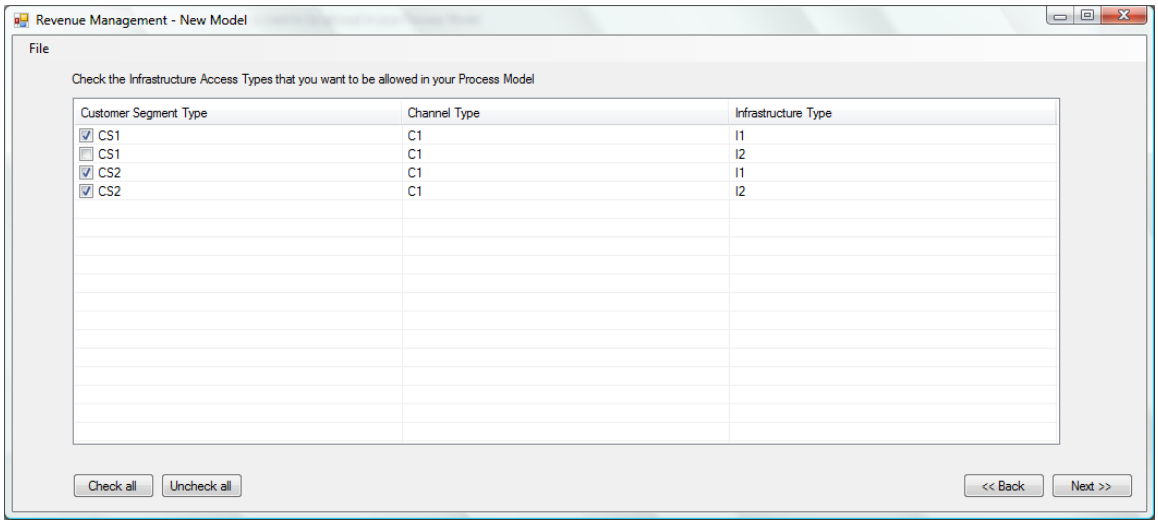


Figura 3-8. Segundo formulario de definición de un Process Model

El botón denominado Back, existente en la esquina inferior derecha del formulario, nos permite volver al primer formulario de definición del modelo y realizar los cambios oportunos. Como ya se ha expuesto en el apartado anterior, estas modificaciones afectan al formulario actual, añadiendo, cambiando, o eliminando las combinaciones de la lista. El botón Next, que se halla junto al botón anterior, permite al usuario acceder al siguiente formulario una vez ha plasmado sus preferencias. El guardado de la información en la base de datos se realiza cuando se pulsa cualquiera de estos dos botones.

El tercer formulario de definición de un modelo de proceso de negocio tiene la función de facilitar al usuario la constatación de que ha elegido los Infrastructure Access Type que realmente deseaba. Por tanto, se muestra en una lista sólo los que se han marcado como válidos en la pantalla anterior. Los botones Back y Next tienen la misma función que el caso anterior, exceptuando la interacción con la base de datos, que en este formulario no se produce por no ser necesaria.

3.4.1.4 FORMULARIO DE DEFINICIÓN DE LOS GENERIC TIMES Y CONSTRAINTS DE UN PROCESS MODEL

El cuarto formulario de definición de un Process Model, cuya apariencia se muestra en la figura 3-9, otorga al usuario la facultad de configurar los Generic Times que tendrán cabida en el modelo, así como las restricciones que los ligan a través de un operador relacional.

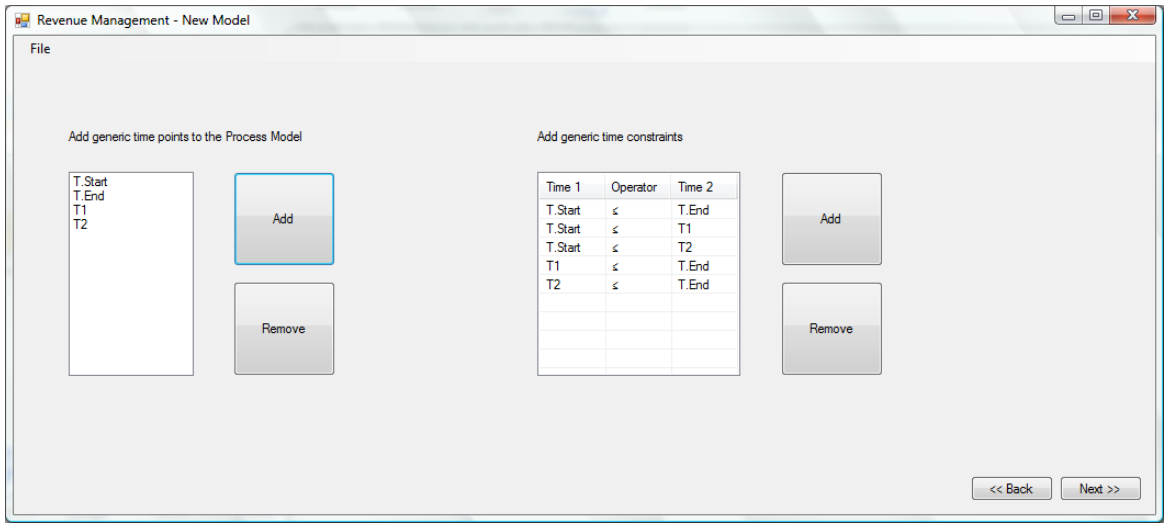


Figura 3-9. Cuarto formulario de definición de un Process Model

En la parte izquierda de la pantalla, una lista muestra los Generic Times, que deben ser como mínimo dos, previamente establecidos por el sistema: T.Start y T.End. Los dos botones anexos permiten añadir o eliminar un Generic Time a excepción de los dos predefinidos.

Por otro lado, en la región derecha formulario, una lista muestra las restricciones o Constraints que ligan los Generic Times. Como mínimo, habrá $2 \cdot n+1$ restricciones, siendo n el número de Generic Times adicionales a los dos preestablecidos. La lista consta de tres columnas, correspondientes al primer término de la restricción (uno de los Generic Times), un operador relacional (en nuestro caso bien el igual, "=", o bien el menor o igual, "≤") y el segundo término de la restricción (otro Generic Time).

Dos botones existentes junto a la lista de Constraints permiten crear y eliminar nuevas restricciones. Si se pincha sobre el botón de creación, surge un cuadro de diálogo que permite elegir cuáles serán cada uno de los tres elementos de la restricción, dando la oportunidad al usuario de elegir en cada caso de entre todas las posibles alternativas. Se muestra en la figura 3-10. Cabe decir que no se puede crear una restricción entre dos elementos que están ligados por una restricción ya existente. Si el usuario lo intenta, el software lo impide y muestra un mensaje informativo. Además, cuando se define una relación del tipo menor o igual, el índice del Generic Time que es menor no puede ser superior al índice del otro Generic Time. De esta forma, por ejemplo, T1 puede ser menor o igual que T2, puede ser igual, o puede no guardar con T2 ningún tipo de relación predefinida, pero en ningún caso T1 puede ser forzosamente superior a T2. Esto es, nunca puede definirse la restricción:

$$T2 \leq T1$$

Antes de pasar al siguiente formulario pulsando en el botón Next, el programa elabora dos matrices que servirán para comprobar que no existen solapes entre Allocation Types que cuentan con idénticos Customer Segment Type, Channel Type y

Value Type, ya que no es posible ofrecer, al mismo tiempo, el mismo tipo de infraestructura, por el mismo tipo de canal al mismo tipo de segmento de cliente y con distinto tipo de valor.

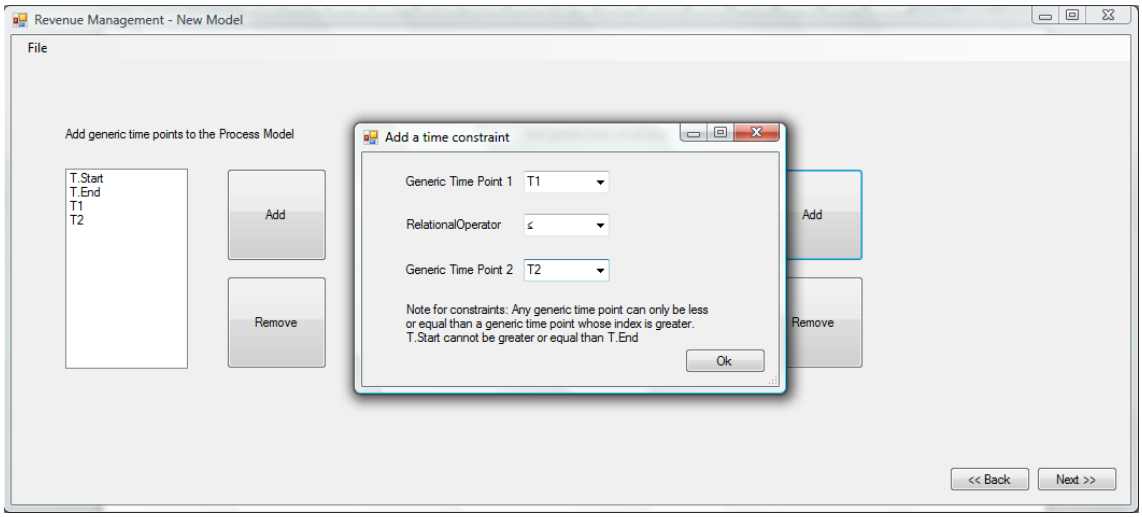


Figura 3-10. Cuadro de diálogo de adición de una restricción temporal

Estas dos matrices son la matriz “de igual” (MatrixEqual) y la matriz “de menor o igual” (MatrixLessOrEqual). Las dimensiones de ambas son n×n, donde n es el número de Generic Times. La fila i y la columna i de cada una de estas matrices están asociadas al Generic Time i. Este índice se relaciona con los Generic Time de la siguiente forma tal y como se puede apreciar en la tabla 3-1.

En MatrixEqual, cada uno de los elementos puede tomar dos valores distintos, 0 ó 1:

- Si un elemento (i, j) cualquiera vale 0, significa que el Generic Time asociado al índice i no está unido mediante una restricción de igualdad al Generic Time ligado al índice j.
- Si un elemento (i, j) cualquiera vale 1, significa que el Generic Time asociado al índice i está unido mediante una restricción de igualdad al Generic Time ligado al índice j.

De forma análoga, en MatrixLessOrEqual, cada uno de los elementos puede tomar dos valores distintos, 0 ó 1.

- Si un elemento (i, j) cualquiera vale 0, significa que el Generic Time asociado al índice i no está unido mediante una restricción del tipo menor o igual al Generic Time ligado al índice j.

- Si un elemento (i, j) cualquiera vale 1, significa que el Generic Time asociado al índice i está unido mediante una restricción del tipo menor o igual al Generic Time ligado al índice j.

Generic Time	**Índice *i***
T.Start	1
T1	2
…	…
Tk	k+1
…	…
T.End	*N*

Tabla 3-1. Relación en los Generic Time y el índice i

El rellenado de ambas matrices se realiza de la siguiente forma:

- En primer lugar, se declaran ambas matrices y se las inicializa, de tal forma que todos los elementos de ambas valen 0.
- Después, se reflejan en la matriz MatrixEqual las restricciones de igualdad definidas por el usuario.
- Posteriormente, se recorre la matriz MatrixEqual de nuevo para tener en cuenta las relaciones de igualdad de segundo orden y sucesivos.
- En cuarto lugar, se inscriben las restricciones del tipo menor o igual en la matriz MatrixLessOrEqual.
- Tras la acción anterior, se tienen en cuenta las relaciones de segundo orden y sucesivos, tanto en la matriz MatrixLessOrEqual como en MatrixEqual, y de forma alterna, n veces, ya que los cambios en una pueden influir en la otra y viceversa.

Este último paso es el más importante y complejo de los cinco. El bucle que se repite n veces, se descompone en cuatro partes diferenciadas. La primera tiene el cometido de corregir la matriz de menor o igual de tal forma que refleje las relaciones representadas en la matriz de igualdad:

```
'i desde 1 hasta n-1, siendo n el número de Generic Times
For i = 1 To lbxTime.Items.Count - 1
```

```
'j desde i+1 hasta n, que junto a i permite recorrer la diagonal
'superior
      For j = i + 1 To lbxTime.Items.Count
      'Si existe igualdad entre el elemento i y el j
            If MatrixEqual(i, j) = 1 Then
            'Se recorren las relaciones de todos los elementos que
            'ligados a cualquiera de los dos por una relación del
            'tipo menor o igual
                  For k = 1 To lbxTime.Items.Count
                  'Se explicita la relación implícita entre el elemento
                  'k y el elemento i o j, según cuál de los dos sea
                  'aquel cuya relación con k no estaba representada en
                  la matriz de menor o igual de forma directa
                        If MatrixLessOrEqual(i, k) = 1 Or _
                        MatrixLessOrEqual(j, k) = 1 Then
                              MatrixLessOrEqual(i, k) = 1
                              MatrixLessOrEqual(j, k) = 1
                        End If
                  Next
                  'Se representa la relación en el sentido opuesto
                  For k = 1 To lbxTime.Items.Count
                        If MatrixLessOrEqual(k, i) = 1 Or _
                        MatrixLessOrEqual(k, j) = 1 Then
                              MatrixLessOrEqual(k, i) = 1
                              MatrixLessOrEqual(k, j) = 1
                        End If
                  Next
            End If
      Next
Next
```

En segundo lugar, se supervisa si es necesario corregir la matriz de relaciones de igualdad:

```
'i desde 1 hasta n-1, siendo n el número de Generic Times
For i = 1 To lbxTime.Items.Count - 1
      'j desde i+1 hasta n, que junto a i permite recorrer la diagonal
      'superior
      For j = i + 1 To lbxTime.Items.Count
      'Se explicita la relación implícita entre los elementos i y j,
      'en caso de que existan dos relaciones entre ambos del tipo menor
      'o igual y en sentidos contrarios
            If MatrixLessOrEqual(i, j) = 1 And MatrixLessOrEqual(j, i)= _
            1 Then
                  MatrixEqual(i, j) = 1
                  MatrixEqual(j, i) = 1
            End If
      Next
Next
```

Posteriormente, se extienden las relaciones de segundo orden y superiores en la matriz de menor o igual:

```
'i desde 1 a n-4, siendo n el número de Generic Times
For i = 1 To (lbxTime.Items.Count - 4)
       'j desde 1 hasta i
       For j = 1 To i
            'Si el elemento indicado vale 1
            If MatrixLessOrEqual(lbxTime.Items.Count - 2 - i, _
            lbxTime.Items.Count - 2 - i + j) = 1 Then
            'Se recorren los elementos de la matriz que hacen referencia
            'al segundo término de la restricción
```

```
                        For k = 1 To 1 + i - j
                        'En caso de que se encuentra una relación
                               If MatrixLessOrEqual(lbxTime.Items.Count - 2 _
                               - i + j, lbxTime.Items.Count - 2 - i + k + j) _
                               = 1 Then
                               'Se extiende al primer elemento de la
                               'restricción
                                      MatrixLessOrEqual(lbxTime.Items.Count _
                                      - 2 - i, lbxTime.Items.Count - 2 - i  _
                                      + k + j) = 1
                               End If
                        Next
                End If
        Next
Next
```

Para explicar qué es exactamente la función de este bucle, se puede utilizar un ejemplo. Se supone que el usuario ha definido seis Generic Times adicionales a T.Start y T.End. Por tanto, la matriz MatrixLessOrEqual cuenta con unas dimensiones de 8×8. La primera fila y la primera columna corresponden a T.Start y la octava fila y la octava columna a T.End. Los elementos intermedios se reparten el resto de filas y columnas de forma coherente. Sabiendo que T.Start es menor o igual que todos los demás Generic Times, que T.End es mayor o igual que el resto de los Generic Times y que además existen dos restricciones adicionales:

$$T3 \leq T4$$
$$T4 \leq T6$$

La matriz MatrixLessOrEqual tiene el siguiente aspecto justo antes de comenzar a ejecutar este último bucle (tabla 3-2):

	T.Start	T1	T2	T3	T4	T5	T6	T.End
T.Start	1	1	1	1	1	1	1	1
T1	0	1	0	0	0	0	0	1
T2	0	0	1	0	0	0	0	1
T3	0	0	0	1	1	0	0	1
T4	0	0	0	0	1	0	1	1
T5	0	0	0	0	0	1	0	1
T6	0	0	0	0	0	0	1	1
T.End	0	0	0	0	0	0	0	1

Tabla 3-2. MatrixLessOrEqual en su estado previo a la propagación de las restricciones implícitas

En la primera ejecución del bucle, i vale 1 y j también. En consecuencia, se evalúa el elemento (5, 6), es decir, (T4, T5). Dado que este tiene un valor de 0, la expresión lógica es falsa y no se ejecuta el código del bloque If. En la segunda ejecución del bucle, i adquiere un valor de 2 y j de 1. Como el elemento (4, 5), esto es, (T3, T4) sí vale 1, la condición lógica se cumple. Cuando el índice k vale 2, la segunda sentencia lógica es verdadera (debido a que el elemento (5, 7) ó (T4, T6) es 1), por lo que, en consecuencia, el elemento (4, 7) ó (T3, T6) adquiere un valor de 1. De esta forma, la relación implícita entre T3 y T6, queda explicitada (tabla 3-3):

$$\left.\begin{matrix} T3 \leq T4 \\ T4 \leq T6 \end{matrix}\right\} \; T3 \leq T6$$

	T.Start	T1	T2	T3	T4	T5	T6	T.End
T.Start	1	1	1	1	1	1	1	1
T1	0	1	0	0	0	0	0	1
T2	0	0	1	0	0	0	0	1
T3	0	0	0	1	1	0	1	1
T4	0	0	0	0	1	0	1	1
T5	0	0	0	0	0	1	0	1
T6	0	0	0	0	0	0	1	1
T.End	0	0	0	0	0	0	0	1

Tabla 3-3. MatrixLessOrEqual tras la propagación de las restricciones implícitas

Finalmente, se propagan las relaciones de segundo orden y superiores en la matriz de relaciones de igualdad, de forma análoga.

3.4.1.5 FORMULARIO DE DEFINICIÓN DE LOS ALLOCATION TYPES DE UN PROCESS MODEL

El quinto y último formulario de definición de un Process Model permite al usuario definir los Allocation Types que formarán parte del modelo. Su aspecto aparece representado en la figura 3-11.

En la parte superior de la pantalla, existe una lista que muestra los Infrastructure Access Types que se han establecido en una fase más temprana del proceso. El usuario tiene la facultad de seleccionar uno de ellos, pulsar el botón Add que se halla junto a la lista y, tras elegir un Value Type, un Generic Start y un Generic End con la ayuda de un cuadro de diálogo, incorporar un Allocation Type basado en el Infrastructure Access Type escogido. La apariencia del cuadro de diálogo puede apreciarse en la figura 3-12.

El nuevo Allocation Type aparece en la lista situada en la mitad inferior del formulario. Justo debajo, se encuentran dos botones cuya función es, respectivamente, editar uno de los Allocation Type o borrarlo. Se actúa sobre el Allocation Type sobre el cual se ha hecho click antes de pulsar sobre el botón correspondiente. Cuando se opta por editarlo, el cuadro de diálogo es el mismo que surge cuando se añade un nuevo Allocation Type.

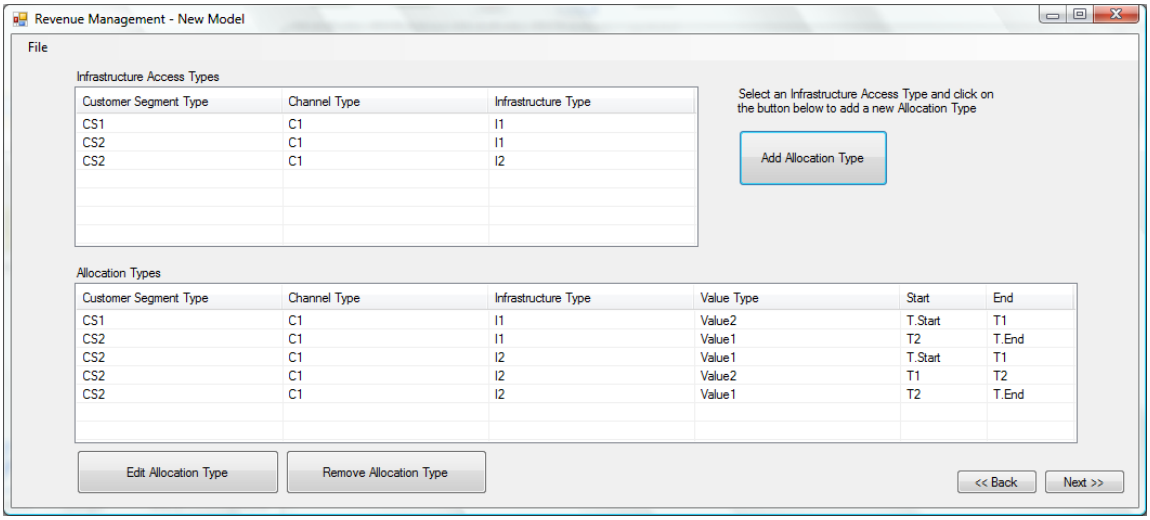


Figura 3-11. Quinto formulario de definición de un Process Model

El guardado de los Allocation Type se produce a través de una rutina denominada SaveAllocationTypes( ), que se ejecuta cuando se pulsa el botón Back, cuando se guarda el Process Model a través del menú File o cuando se pulsa el botón Next y los Allocation Types definidos cumplen con la condición necesaria para dar por finalizada la definición completa del Process Model.

Esta condición dicta que dos Allocation Types basados en los mismos Customer Segment Type, Channel Type e Infrastructure Type, no pueden solaparse en un intervalo genérico de tiempo. En el caso de que no sea seguro el solape, pero sí posible, el programa rechazará igualmente el cierre de la definición del modelo.

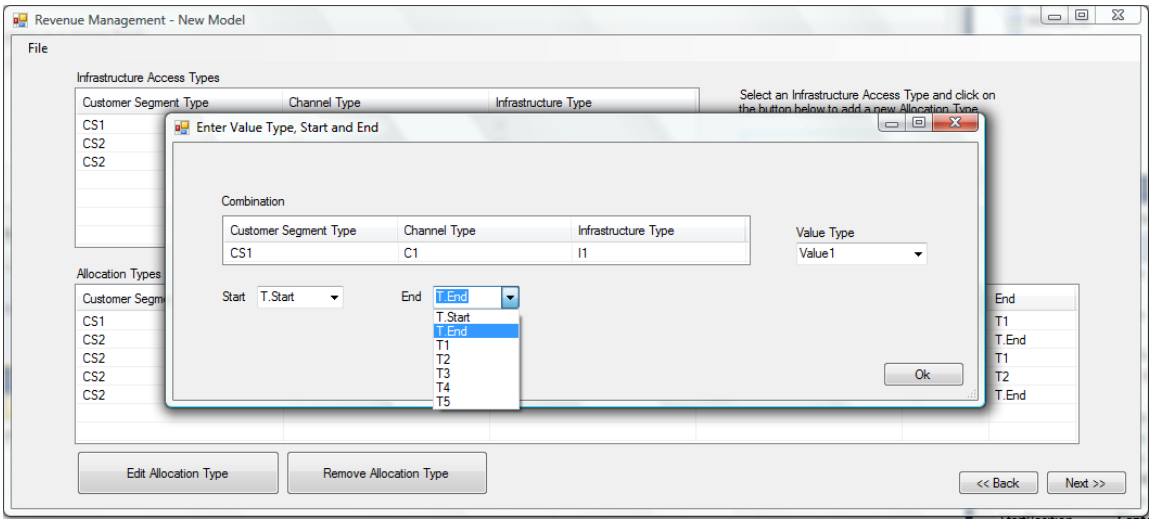


Figura 3-12. Cuadro de diálogo de configuración de un Allocation Type

El algoritmo cuya función es comprobar si se cumple la condición es el siguiente:

```
'Si el comienzo del Allocation Type 1 es menor o igual que el comienzo del
'Allocation Type 2 y el comienzo del Allocation Type 2 no es menor o igual
'que el comienzo el Allocation Type 1
If frm04NewModel.MatrixLessOrEqual(Start1, Start2) = 1 And _
frm04NewModel.MatrixLessOrEqual(Start2, Start1) = 0 Then
       'Si el final del Allocation Type 1 es menor o igual que el comienzo
       'del Allocation Type 2
       If frm04NewModel.MatrixLessOrEqual(End1, Start2) = 1 Then
              Correct2 = Correct2 'Correcto
       Else
              Correct2 = 0 'Incorrecto
              'Se guarda el grupo de Allocation Types donde está el error
              GroupError = i
       End If
'Si no, si el comienzo del Allocation Type 1 es igual al comienzo 'del
'Allocation Type 2
ElseIf frm04NewModel.MatrixEqual(Start1, Start2) = 1 Then
       'Si, el final del Allocation Type 1 es igual al comienzo del
       'Allocation Type 2
       If frm04NewModel.MatrixEqual(End1, Start2) = 1 Or
       frm04NewModel.MatrixEqual(Start1, End2) = 1 Then
              Correct2 = Correct2 'Correcto

       Else
              Correct2 = 0 'Incorrecto
              'Se guarda el grupo de Allocation Types donde está el error
              GroupError = i
       End If
'Si no, si el comienzo del Allocation Type 2 es menor o igual al comienzo
'del Allocation Type 2 y el comienzo del Allocation Type 1 no es menor o
'igual que el comienzo del Allocation Type 2
ElseIf frm04NewModel.MatrixLessOrEqual(Start2, Start1) = 1 And
frm04NewModel.MatrixLessOrEqual(Start1, Start2) = 0 Then
       'Si el final del Allocation Type 2 es menor o igual al comienzo del
       'Allocation Type 1
```

```
        If frm04NewModel.MatrixLessOrEqual(End2, Start1) = 1 Then
               Correct2 = Correct2 'Correcto
        Else
               Correct2 = 0 'Incorrecto
               'Se guarda el grupo de Allocation Types donde está el error
              GroupError = i
        End If
Else
        Correct2 = 0 'Incorrecto
        'Se guarda el grupo de Allocation Types donde está el error
       GroupError = i
End If
```

Si todos los Allocation Types superan con éxito el corte anterior, aparece un cuadro de diálogo (figura 3-13) que permite al usuario guardar el modelo y continuar con la definición de una Process Instance, simplemente guardar el modelo y volver a la pantalla inicial, o bien descartarlo, borrándolo de la base de datos, y regresar también a la pantalla inicial.

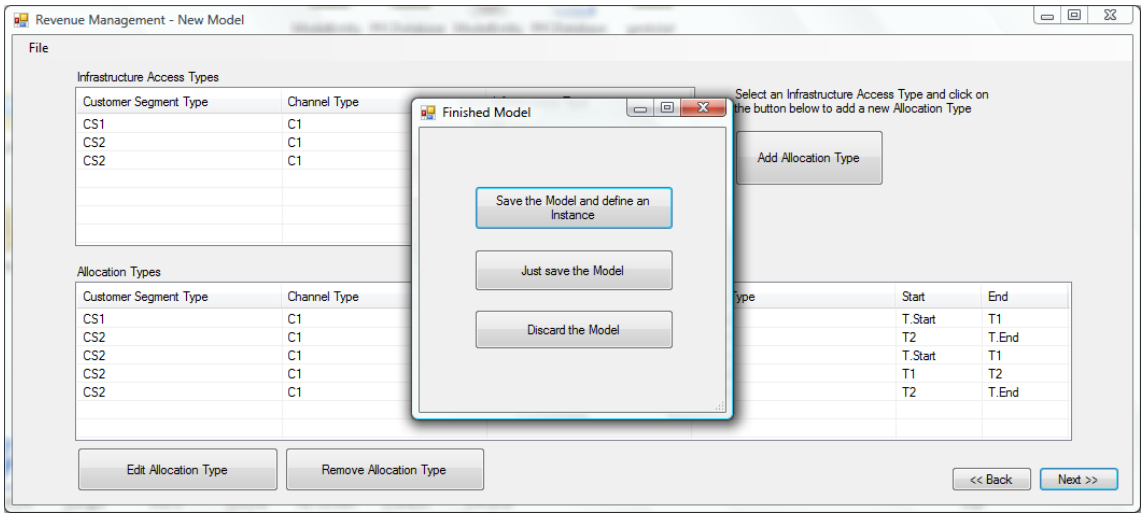


Figura 3-13. Cuadro de diálogo de fin de definición de un Process Model

4

MÓDULO DE DEFINICIÓN DE ATRIBUTOS

En el presente capítulo, que sigue la misma estructura que el capítulo anterior, se presenta, en primer lugar, el diseño conceptual del modelo de atributos dentro del marco de una instancia de un proceso de negocio. La segunda parte del capítulo corresponde al modelo de datos, donde se analiza el diseño de la base de datos que procura soporte a la aplicación, incluyendo un diagrama explicativo de la misma en IDEF1X. El diseño de la aplicación, junto a la justificación de su estructura y el diagrama de estados correspondiente, se trata en la tercera parte. Por último, se expone cómo se ha desarrollado la implementación del módulo del software correspondiente a la configuración de atributos en una instancia de un proceso de negocio.

4.1 DISEÑO CONCEPTUAL

El presente módulo es el que da inicio a la definición de una instancia de proceso de negocio. El nivel de Process Instance se corresponde con el tercer nivel jerárquico del modelo de partida, cuyas dos primeras capas las integraban el metamodelo y el modelo de procesos. Sin embargo se han realizado algunas modificaciones sobre el modelo de partida a nivel de instancia de proceso (ver figura 4-1).

La principal diferencia reside en la inclusión de la clase entidad de la instancia. En cada instancia de proceso de negocio (Business Process Instance) existen varias entidades de la instancia (Instance Entities), que son a su vez instancias de las entidades del modelo (Model Entities). Son los elementos básicos que se relacionan en el marco de la instancia de proceso y cuyo significado hemos explicado uno a uno con anterioridad en el capítulo 2. Las entidades de la instancia pueden clasificarse de forma disjunta y completa como segmento de cliente (Customer Segment), canal (Channel), conjunto de infraestructura (Infrastructure Set), valor (Value), acceso a una infraestructura (Infrastructure Access), tiempo (Time) y asignación (Allocation). Es decir, toda entidad de la instancia pertenece forzosamente a uno de esos tipos y sólo a uno. La relación entre la entidad de la instancia y la entidad del modelo es una generalización de la relación de instanciación entre los elementos de la enumeración anterior y sus padres.

En el diagrama de clases UML de la definición de un Process Instance (Figura 4-1), se incluye un reducido número de atributos como pertenecientes a las clases de este tercer nivel, tal y como se puede apreciar en la figura. Los atributos que aparecen representados son algunos de los que se consideran más significativos en el proceso de definición de una instancia de un proceso de negocia de asignación de infraestructuras y pueden ser útiles de cara a la caracterización de los tipos de evento en el siguiente módulo.

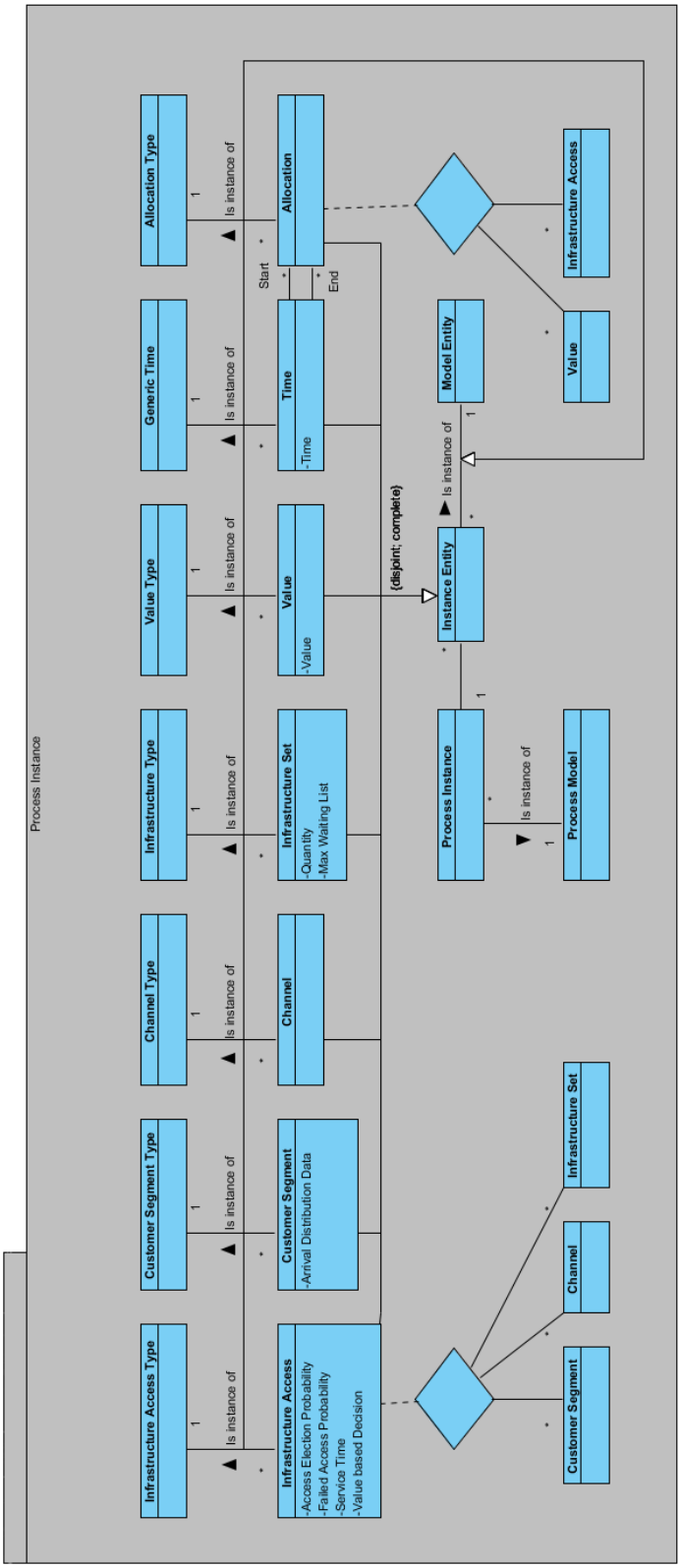


Figura 4-1. Diagrama de clases UML de Process Instance, actualizado con respecto al modelo de partida

Estos atributos, según a la clase a la que pertenecen son:

- Atributo de Customer Segment:
  - Arrival Distribution Data: Se corresponde con la distribución de llegadas al sistema de los clientes correspondientes a ese segmento. El acceso se produce instantáneamente, al mismo tiempo de su llegada al sistema. Sólo puede actuar como parámetro, cuyo valor es fijado por el usuario, no como variable.

- Atributos de Infrastructure Set:
  - Quantity: Es la cantidad de elementos de infraestructuras de un tipo concreto de las que se dispone para ser asignados a los clientes. Puede ser un parámetro fijado por el usuario o una variable. En ese último caso, el usuario puede utilizar el modelo como parte de un análisis que dé soporte a una decisión de dimensionamiento de sus instalaciones o recursos, de carácter estratégico o táctico.
  - Max Waiting List: Es el número máximo de clientes que se permite que permanezcan en la lista de espera para acceder a la asignación definitiva de una infraestructura de un determinado tipo. En caso de que el usuario decida que no va a incluir una lista de espera en su modelo de proceso de negocio, basta con que asigne a este atributo un valor de 0 o bien directamente lo elimine.

- Atributo de Value:
  - Value: Es el valor propiamente dicho que va asociado a un Allocation determinado. Este valor puede referirse simplemente a las unidades monetarias recibidas por la asignación del recurso, o bien ser sensible a otro tipo de consideraciones no estrictamente financieras, como sucede en las aplicaciones del Revenue Management al Sector Sanitario.
    Este atributo puede actuar bien como parámetro o bien como variable. En ese último caso, permitiría, en el ejemplo de un hotel que utilizase esta herramienta, fijar el precio que va asociado a una determinada Allocation, de tal forma que se maximice el ingreso obtenido.

- Atributo de Time:
  - Time: Es el tiempo concreto expresado en unidades temporales que se le asigna a un Time. Este atributo puede ser parámetro o variable, excepto en el caso del primer y último Time, que van a marcar los momentos de inicio y de fin (Start y End) y son parametrizados por el usuario de forma obligatoria.

- Atributos de Infrastructure Access:
  - Access Election Probability: Es la probabilidad de que un cliente que accede al sistema opte por un determinado Infrastructure Access.
  - Value based Probability: Es la probabilidad de que un cliente acepte la oferta de asignación en función del valor de la misma.

- Failed Access Probability: Es la probabilidad de que el acceso de un cliente, una vez ha aceptado la oferta, resulte fallido y no se concrete en una asignación.
- Service Time: Es el tiempo de servicio asociado a un determinado Infrastructure Instance.

Contando con los atributos que se acaban de desarrollar y continuando con el ejemplo del establecimiento de alquiler de automóviles del capítulo anterior, las instancias de Infrastructure Set podrían tener el siguiente aspecto en una de las instancias del proceso de negocio (tabla 4-1):

Process Model	Process Instance	Infrastructure Set	Quantity	Max Waiting List
Process Model 1	Process Instance 1	Coche compacto	10	3
Process Model 1	Process Instance 1	Coche de alta gama	3	1

Tabla 4-1. Instancias de Infrastructure Set en el ejemplo del establecimiento de alquiler de automóviles

Dada la gran diversidad de problemas de asignación de infraestructuras que pueden presentarse, es seguro que en un momento dado el usuario decidirá que resulta necesario incluir más atributos para definir su instancia de proceso y solucionar su problema de optimización, sea cual sea la alternativa por la que va a optar a la hora de resolverlo. Por ello, es necesario contar con un modelo de atributos que le permita utilizar sus propios parámetros y variables, aquellos que considere oportunos para la correcta definición de su modelo. Los atributos que hemos descrito anteriormente encajan a la perfección dentro de este modelo de atributos.

Un Attribute (Atributo) es una característica que le es propia a todas las Instance Entity de una instancia de proceso, si bien en cada caso esa característica puede adquirir un valor distinto. Por ejemplo, sirviéndonos de los atributos explicados en el apartado anterior, en una determinada instancia de un proceso todos los Customer Segments pueden tener en común que cuentan con una determinada distribución de llegadas, si bien el tipo de distribución y el valor de los parámetros de la misma pueden ser distintos para cada Customer Segment. Cada Attribute se relaciona con uno o más parámetros de atributo Attribute Parameters (Parámetros de Atributo). Estos parámetros de atributo son los que pueden adquirir un valor concreto. Para explicar la clasificación de la clase Attribute Parameter, es necesario exponer previamente las que corresponden a la clase Attribute.

Como se aprecia en la figura 4-2, El modelo de atributos se articula en torno a tres clasificaciones de la clase Attribute (Atributo). Cada clasificación, por separado es completa y disjunta. Esto es, cada atributo pertenece siempre a una de las categorías de la clasificación y sólo a una.

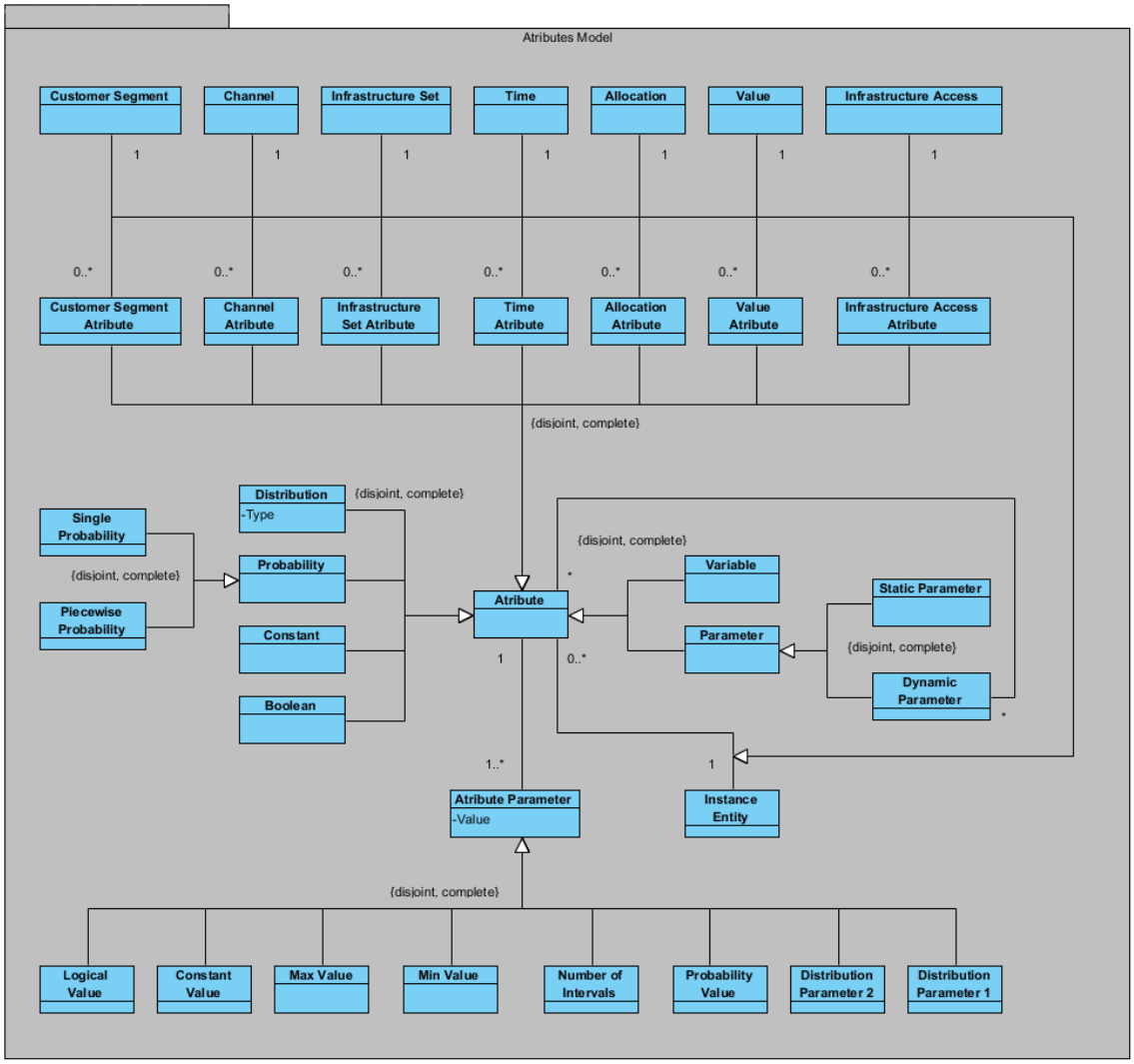


Figura 4-2. Diagrama de clases UML del modelo de Atributos

En primer lugar, los atributos pueden clasificarse en función de su asociación a un elemento perteneciente a una de las siguientes clases del tercer nivel del modelo jerárquico (Business Process Instance): Customer Segment, Channel, Infrastructure Set, Time, Value, Infrastructure Access y Allocation. A cada elemento de cada una de estas clases le corresponderá ninguno, uno o varios atributos.

En segundo lugar, un atributo es de uno de los siguientes tipos: distribución estadística (Distribution), probabilidad (Probability), constante (Constant) o booleano

(Boolean). En el caso de las distribuciones, se deberá indicar su tipo en el momento de definición del atributo, que se almacena en Type, que es el atributo de un atributo.

A priori, se puede elegir cualquier tipo de distribución: normal, exponencial, binomial, Poisson, etc. No obstante, es importante tener en cuenta si la distribución elegida se encuentra en la biblioteca de distribuciones estadísticas de las herramientas que se van a utilizar tanto para la construcción del modelo como para su posterior resolución. Los atributos de tipo probabilidad pueden a su vez clasificarse (de forma disjunta y completa) como atributos de probabilidad única (Single Probability) o como atributos de probabilidad definida a trozos (Piecewise Probability). Este último tipo nos permite considerar distintos valores probabilísticos según el tramo en el que se evalúe, tal y como se aprecia en la figura. Un atributo de tipo Constant se identifica con una constante numérica, así como uno de tipo Boolean se lo hace con un valor lógico de verdadero o falso.

Por último y en tercer lugar, un atributo puede actuar como parámetro (Parameter) o como variable (Variable). A su vez, los parámetros, a través de una clasificación disjunta y completa, pueden ser dinámicos (Dynamic Parameter) o estáticos (Static Parameters). La diferenciación entre variables y parámetros se realiza en función de cuáles son los atributos cuyo valor de parámetro de atributo permitirá optimizar la función objetivo, y que por tanto se fijará en la fase de resolución del modelo (variables), y cuáles son aquellos cuyos valores de sus parámetros de atributo asociados se definen de antemano por parte del usuario, a modo de restricción (Static Parameter) o bien el valor de su parámetro de atributo es cambiante en tiempo de resolución en base a una serie de reglas que lo relacionan con otros atributos (Dynamic Parameters).

Un atributo no participa de estas tres clasificaciones de manera independiente, sino que su taxonomía en lo que respecta a una de las clasificaciones puede condicionar su pertenencia a uno u otro tipo en otra. En concreto, las dos clasificaciones que expresan incompatibilidades son las que aparecen a izquierda y derecha de la clase Attribute en la figura. Por ejemplo, un atributo de tipo distribución estadística no podrá ser variable. En general, las compatibilidades entre estas dos clasificaciones se expresan en la tabla 4-2.

Clasificación 1	Clasificación 2
Distribution	Parameter
Probability	Parameter
Constant	Variable
Constant	Parameter
Boolean	Variable
Boolean	Parameter

Tabla 4-2. Compatibilidad de clasificaciones de Attributes

Los parámetros de atributo (Attribute Parameters) se pueden clasificar de forma disjunta y completa en:

- Logical Value: Valor lógico, sólo puede ser True o False. Está asociado a los atributos de tipo Boolean.
- Constant Value: Valor numérico que se relaciona con los atributos clasificados como Constant.
- Distribution Parameter 1: Parámetro numérico de un atributo del tipo Distribution.
- Distribution Parameter 2: Análogo al anterior.
- Probability Value: Valor probabilístico, comprendido entre 0 y 1. Asociado tanto a atributos de los tipos Single Probability y Piecewise Probability.
- Max Value: Valor numérico que indica cuál es el extremo superior del rango en el que está definido un atributo de probabilidad a trozos (Piecewise Probability).
- Min Value: Valor numérico del extremo inferior del rango en el que está definido un atributo de probabilidad a trozos (Piecewise Probability)
- Number of intervals: Es el valor numérico que indica cuál es el número de tramos o intervalos ligados a un atributo del tipo Piecewise Probability.

Los distintos tipos de parámetros de atributos se relacionan con sus atributos correspondientes de manera sencilla, excepto en el caso de aquellos asociados a un atributo Piecewise Probability. Para facilitar la comprensión de este caso concreto, se recurre a un ejemplo. Suponemos que los valores de los distintos parámetros de atributo asociados a un mismo atributo del tipo Piecewise Probability son los que aparecen en la tabla 4-3:

Min Value	Max Value	Number of Intervals	Probability Value	Probability Value	Probability Value
0	90	3	0,75	0,65	0,40

Tabla 4-3. Ejemplo de parámetros de un atributo del tipo PiecewiseProbability

La información contenida en la tabla 4-3 puede sintetizarse, tal y como se expone en la tabla 4-4:

Interval	Probability Value
0 – 30	0,75
30 – 60	0,65
60 -90	0,40

Tabla 4-4. Ejemplo de sintetización de la información contenida en los parámetros de un atributo del tipo Piecewise Probability

En la fase del diseño del modelo, se contempló la posibilidad de utilizar los parámetros de atributo que se muestran en la tabla 4-4 en lugar de los que se muestran en la tabla 4-3. Sin embargo, se desechó esta idea debido a que dificultaría el tratamiento de los datos por parte de una aplicación. La información anterior puede expresarse gráficamente, tal y como se aprecia en la figura 4-3.

Cuando se vaya a utilizar un atributo del tipo Piecewise Probability, es necesario combinarlo con otro atributo que determine el valor en el que se evalúa la probabilidad definida a trozos. Esta combinación se puede modelizar a través de la relación muchos a muchos de un atributo con otro atributo del tipo Dynamic Parameter.

Se puede explicar adecuadamente con un ejemplo. El atributo Value based Decision se clasifica simultáneamente como Infrastructure Access Attribute, Piecewise Probability y Static Parameter. Expresa la probabilidad de que un cliente que pertenece a un determinado segmento de cliente y que accede a una oferta de asignación de infraestructuras a través de un canal concreto, acepte la oferta. Dicha probabilidad depende del valor asociado a esa asignación (normalmente se trata del precio), y se define a trozos por parte del usuario. El atributo Value based Decision se relaciona con el atributo de tipo Dynamic Parameter y booleano denominado Request, que pertenece a Allocation. A su vez, Request también se asocia al atributo Value de la clase Value. El parámetro de atributo del tipo Constant Value de este último será el encargado de seleccionar el tramo en el cual se evalúa la probabilidad de que el cliente acepte la oferta. El resultado de esa evaluación es el que determina el valor True o Falso del Attribute Parameter de tipo Boolean Value perteneciente a Request. La relación existente entre estos tres atributos se encontrará predefinida en el software, de tal forma que el usuario se consciente de su existencia y pueda definir relaciones similares con atributos de su propia creación.

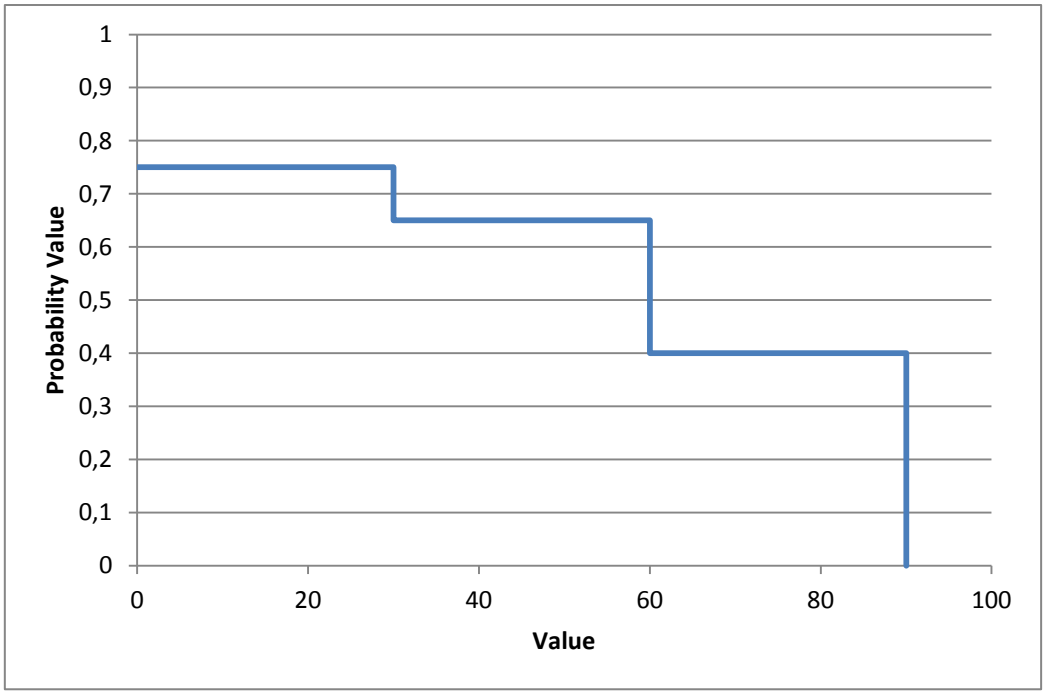


Figura 4-3. Ejemplo de probabilidad definida a trozos

Cabe mencionar, que no todo atributo del tipo Dynamic Parameter se relaciona con otros miembros de la clase Attribute, normalmente debido a que el procedimiento para determinar el valor del parámetro de atributo asociado implica a otros atributos pertenecientes a un nivel de instanciación mayor, el nivel que modela la ejecución de un modelo de proceso de negocio. Sin embargo, estos atributos de tipo Dynamic Parameter se modelan en el nivel Process Instance porque resultan necesarios para definir el modelo dinámico del Process Instance, que será objeto de análisis en el siguiente capítulo.

4.2 MODELO DE DATOS

En esta sección se expone el modelo de datos de este módulo. Para ello, no serviremos del diagrama IDEF1X elaborado con ERWin que aparece en la figura 4-4, y en el cual puede apreciarse el diseño lógico del segmento de la base de datos correspondiente al modelo de atributos y parcialmente a la definición de un Process Instance.

En lo que respecta al diseño de modelo de datos, la primera decisión importante consiste en decidir qué tablas son necesarias. Se considera cada una de las entidades que aparecen en los diagramas de clases UML representados en las figuras 4-1 y 4-2. Todas requieren una tabla propia en la base de datos, excepto las que son subtipos de Attribute y de Attribute Parameter. Esta decisión se debe a que los registros de estas tablas

pueden identificarse fácilmente como pertenecientes a un subtipo con la ayuda de la información contenida en un atributo, TypeOfAttribute y Type_ respectivamente.

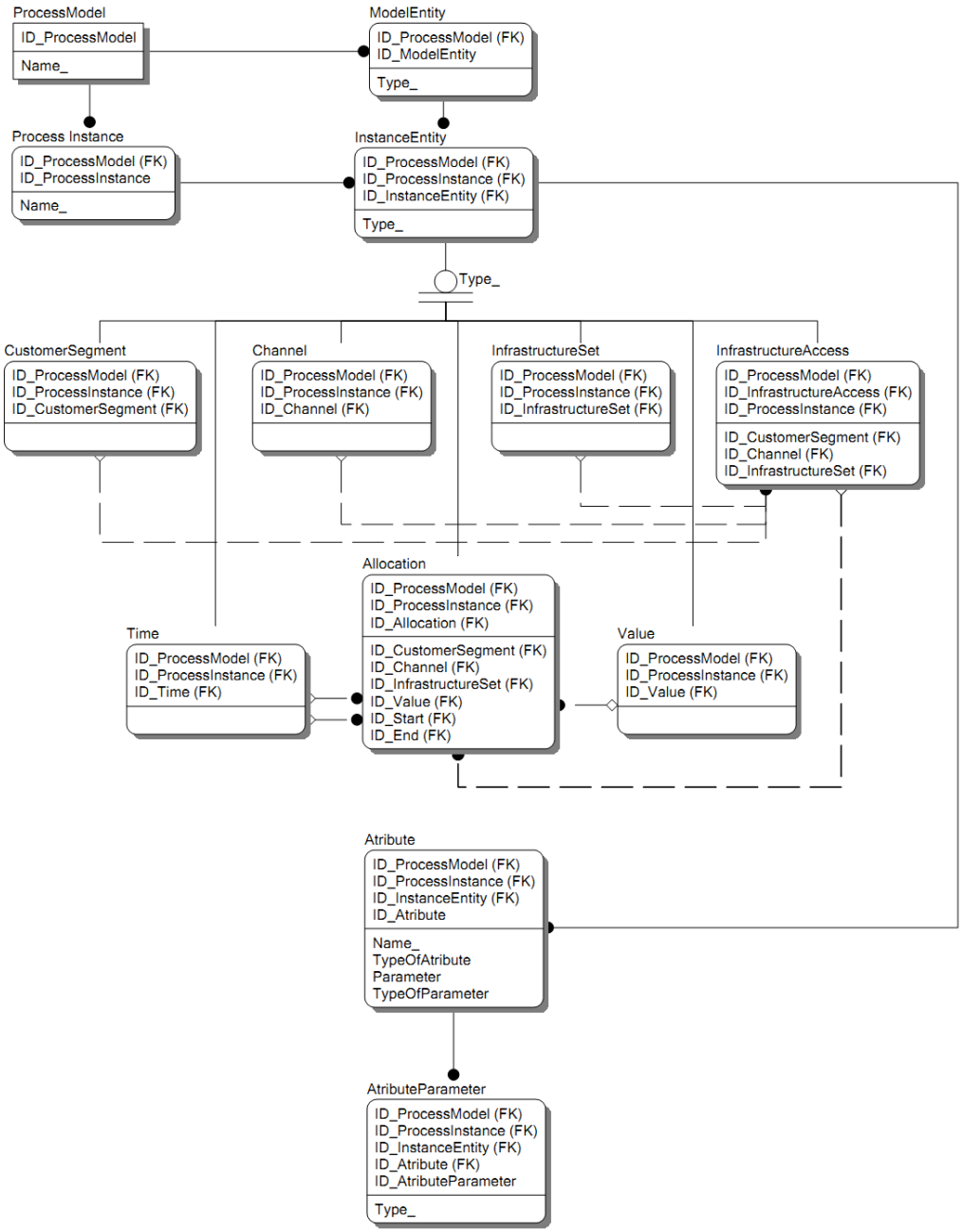


Figura 4-4: Diagrama IDEF1X del segmento de la base de datos correspondiente al modelo de Atributos

Así pues, las tablas necesarias son: ProcessModel, Process Instance, ModelEntity, InstanceEntity, CustomerSegment, Channel, Infrastructure Set, InfrastructureAccess, Value, Time, Allocation, Attribute y AttributeParameter, tal y como puede apreciarse en la figura anterior.

La tabla en la cual se anota el primer registro cuando se crea una instancia de proceso de negocio es ProcessInstance. La clave primaria de esta tabla es numérica y en parte heredada de ProcessModel y en parte propia. Además la tabla cuenta con un campo adicional, Name_, un atributo que se corresponde con el nombre que el usuario decide darle al Process Instance. El guión bajo del final es necesario porque "Name" es una palabra clave del lenguaje SQL.

Por coherencia, fue necesario introducir en el nivel del Process Instance la clase Instance Entity, instancias directas de las entidades básicas presentes en el proceso de modelo de negocio. Pueden ser de distintos tipos de forma exclusiva y excluyente: Customer Segment, Channel, Infrastructure, Value, Time, Infrastructure Access o Allocation. Cada Instance Entity se asocia a un Process Instance concreto. La clave de la tabla InstanceEntity es una combinación de claves ajenas: ID_ProcessModel, ID_ProcessInstance e ID_InstanceEntity, que si bien se le ha cambiado el nombre, se trata de la clave ID_ModelEntity de la tabla ModelEntity. Esta última decisión se debió a que en un mismo Process Instance sólo se utiliza una instancia de cada ModelEntity, por lo que no tiene sentido otorgarle una clave propia a las InstanceEntity. Además, el atributo Type_ describe el tipo de Instance Entity al cual pertenece el registro, es decir, actúa como discriminador en la clasificación. En consecuencia, en el diagrama IDEF1X aparece junto al símbolo de la relación de categorización, que como hemos indicado anteriormente, es un circulo blanco, con una o dos líneas debajo según la completitud de la clasificación. De esta forma, cuando es necesario cargar una instancia de proceso de negocio desde la base de datos, el proceso es mucho más intuitivo y sencillo.

En principio pudiera pensarse que no es necesario implementar las tablas CustomerSegment, Channel, Infrastructure ni Value, ya que cuando se requiriese la información correspondiente a las entidades asociadas a dichas tablas, lo cual sucede cada vez que el usuario carga o define una instancia, ésta podría reconstruirse fácilmente a partir de los registros de la tabla ModelEntity. Sin embargo, con vistas a no dificultar la expansión de los diferentes niveles del modelo, se ha decidido dotar a la base de datos con las tablas mencionadas con anterioridad. La razón es que, en un momento dado, podría resultar conveniente asignar a alguna o a varias de estas entidades atributos propios o que no tuvieran sentido para las demás. Si bien es cierto que sería posible añadir tantos atributos extra en la tabla InstanceEntity como fuera necesario y permitir en ellos valores nulos, lográndose el mismo efecto práctico, no sería la solución más elegante desde el punto de vista conceptual. Así mismo, dado que en el módulo de definición de un modelo de proceso de negocio se han implementado las tablas CustomerSegmentType, ChannelType, InfrastructureType y ValueType, resulta coherente hacer lo mismo en el nivel de la instancia de proceso de negocio.

De manera análoga a lo que sucedía en el nivel superior, los diferentes CustomerSegment, Channel, Infrastructure y Value están identificados por medio de una clave primaria formada por una clave numérica ajena heredada (ID_InstanceEntity, que pasa a denominarse ID_CustomerSegmentType, ID_ChannelType, ID_InfrastructureType e ID_ValueType, según el caso), y las claves ajenas del Process Model y del Process Instance correspondientes, heredadas a través de la misma relación.

Cada registro de la tabla InfrastructureAccess cuenta con una clave primaria formada por una clave ajena, ID_InfrastructureAccess, que coincide con su clave en la tabla InstanceEntity (ID_InstanceEntity) combinada con una clave ajena correspondiente a su Process Model (ID_ProcessModel) y otra asociada a su Process Instance (ID_ProcessInstance), heredadas a través de InstanceEntity. Cada registro de esta tabla se corresponde con una combinación de un Customer Segment, un Channel y un Infrastructure Set en el contexto de unos determinados Process Model y Process Instance. Es decir, una instancia de la entidad Infrastructure Access, que hereda como claves no primarias las claves ajenas de los Customer Segment, Channel e Infrastructure Set correspondientes.

Un registro de la tabla Allocation se corresponde con una combinación de un Customer Segment, un Channel y un Infrastructure Set, ligada a su vez a un determinado Value, que estará vigente entre dos puntos temporales o Times (Start y End) de un determinado Process Instance. La clave primaria de un registro de la tabla Allocation es la clave numérica ajena ID_Allocation (que coincide con su clave ID_InstanceEntity en la tabla InstanceEntity) combinada con las claves del Process Model y del Process Instance a los cuales está asociado ese Allocation, heredada a través de la relación de categorización con la entidad Instance Entity. Además, cada registro cuenta como claves no primarias con las claves ajenas de los Customer Segment, Channel, Infrastructure Set, Value, Start y End correspondientes.

Los registros de la tabla Time se corresponden con tiempos genéricos que actúan como momentos de inicio y de fin de vigencia de los distintos Allocation. Su clave está formada por la clave ajena ID_Time, de tipo numérico y que coincide con la clave ID_InstanceEntity correspondiente en la tabla InstanceEntity, y las claves ajenas ID_ProcessModel e ID_ProcessInstance del modelo e instancia de proceso de negocio respectivamente a los cuales pertenece ese genérico, heredadas a través de su relación de categorización con InstanceEntity. En todo Process Instance completamente definido existen, como mínimo, dos Times, denominados T.Start y T.End, que se asimilan con los tiempos de inicio y fin de actividad de la instancia de proceso de negocio que se pretende modelar.

La tabla Attribute tiene una clave principal compuesta por tres claves heredadas y una clave propia. Las claves heredadas son ID_ProcessModel, ID_ProcessInstance e ID_InstanceEntity, todas ellas recibidas a través de su relación con la tabla InstanceEntity. La clave propia es ID_Attribute, que identifica de manera unívoca a cada uno de los atributos de una entidad concreta, en una instancia de un determinado modelo de proceso de negocio. El atributo Name_ sirve para almacenar el nombre del

atributo, que además, para facilitar su identificación, se ha decidido que sea único. El software deberá velar por que no se viole esta condición.

Por otra parte, TypeOfAttribute es un atributo que nos permite identificar ante qué tipo de atributo estamos desde el punto de vista de la segunda clasificación que se ha expuesto en el apartado de diseño conceptual de este mismo capítulo. Así pues, un atributo es de uno de los siguientes tipos: distribución estadística (Distribution), probabilidad (Probability), constante (Constant) o booleano (Boolean), por lo que el atributo TypeOfAttribute, de tipo texto, debe tomar uno de esos cuatro valores.

Tanto Parameter como TypeOfParameter, el atributos de la tabla Attribute, hacen referencia a la tercera clasificación de la clase Attribute, tal y como se desarrolló en el diseño conceptual del módulo de Atributos. En primer lugar, Parameter, tomando un valor de “YES” o “NO”, nos permite saber si estamos ante un parámetro o una variable, respectivamente. En caso de que se trate de un parámetro, el atributo TypeOfParameter, de tipo texto, nos permitirá saber si es estático (Static) o dinámico (Dynamic).

Finalmente, la tabla AttributeParameter almacena los registros correspondientes a los parámetros de los atributos que aparecen en la tabla Attribute. Dispone de una clave principal compuesta por cuatro claves heredadas y una clave propia. Las claves heredadas son ID_ProcessModel, ID_ProcessInstance, ID_InstanceEntity e ID_Attribute, todas ellas recibidas a través de su relación con la tabla Attribute. La clave propia es ID_AttributeParameter, que identifica de manera unívoca a cada uno de los parámetros de un determinado atributo, en una instancia específica de un modelo de proceso de negocio. El atributo Type_, de tipo texto, nos permite identificar ante qué tipo de parámetro de atributo nos encontramos, siguiendo la clasificación de los mismos que se ha expuesto en el capítulo anterior.

4.3 DISEÑO DE LA APLICACIÓN

En esta sección se expone el proceso de diseño del módulo la aplicación que da soporte a la definición de los atributos de una instancia de proceso. Se desarrolla cuál es la estructura y se muestra un diagrama de secuencia de UML (figura 4-5).

Cuando el usuario decide que va a crear una nueva instancia de proceso, el programa debe solicitarle un nombre de la instancia y además asegurarse de que sea único, para que no haya posibilidad de confusión con otras instancias del mismo modelo. Se descarta la posibilidad de que el usuario pueda introducir el nombre de una instancia ya existente y que el programa le permita sobrescribir. El sentido de esa decisión se debe no se desea correr el riesgo de que se produzcan pérdidas involuntarias de información valiosa o cuya introducción ha sido muy laboriosa.

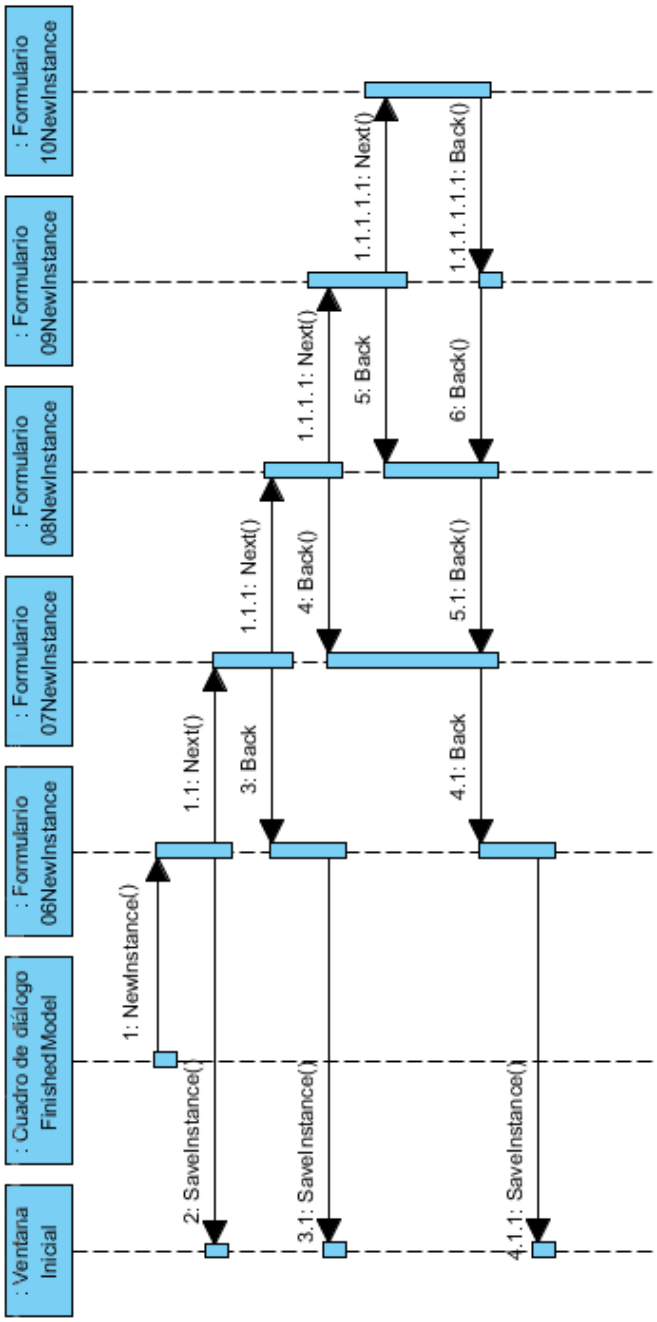


Figura 4-5. Diagrama de secuencia en UML del módulo de la aplicación asociado a la definición de atributos

La creación de una instancia y de la definición de los atributos presentes en la misma no requiere en absoluto que el programa haga referencia directa a las instancias de cada una de las piezas básicas del modelo conceptual, es decir, a las instancias de las Instance Entities que forman la base y son: Customer Segment, Channel, Infrastructure Set, Value, Time, Infrastructure Access y Allocation. Se debe a que los atributos que se definen se asignan a todas las instancias, sin excepción, de alguna de las clases que acabamos de listar. Por ejemplo, si decidimos que Arrival Distribution Data es un

atributo necesario en nuestro modelo, bastará con asociarlo, al menos en el nivel de aplicación, con Customer Segment, no con cada uno de las instancias de la clase Customer Segment establecidas por el usuario durante el uso del módulo de la aplicación correspondiente a la definición de un modelo de negocio.

Por ello, se decide que en el primer formulario, junto al nombre del modelo de proceso y de la instancia de proceso, existan siete listas, cada una de ellas correspondiente a los atributos ligados a cada una de las siguientes siete entidades: Customer Segment, Channel, Infrastructure Set, Value, Time, Infrastructure Access y Allocation. Los elementos que forman parte de estas listas deben ser modificables por parte del usuario, por lo que por medio de tres botones colocados junto a cada lista se le permite añadir un nuevo atributo a la lista, editarlo o eliminarlo de la instancia de proceso de negocio. Además, el formulario debe contar con un botón que permita al usuario pasar al siguiente formulario.

El segundo formulario de definición de un modelo de proceso debe permitir al usuario elegir cuál es el atributo que será la variable de optimización en esta instancia del proceso. Sólo uno de ellos puede ser variable, y tiene que ser como mínimo uno. En este caso, cuando se muestran los atributos, lo cual se puede hacer en una lista, no basta con que se sepa a qué clase de Instance Entity pertenecen, sino que es necesario conocer a qué instancia concreta de Instance Entity corresponden. Retomando el ejemplo del establecimiento de alquiler de automóviles, si decidimos que la variable va a ser un atributo de Time, necesitamos concretar si se trata del valor de T.Start, de T1, de T2, o de T.End. Para permitir esta identificación del atributo y de la instancia concreta de Instance Entity a la que corresponde, la lista debe incluir no sólo el nombre del atributo, su tipo y el tipo de la entidad a la que va asociado, sino también otros campos que según el tipo de entidad de que se trate, se utilizarán o no: Customer Segment, Channel, Infrastructure Set, Value y dos campos denominados Time. El hecho de incluir estos dos campos se debe a una cuestión de versatilidad: nos permiten representar tanto los atributos correspondientes a Time como los que pertenecen a Allocation con el mínimo número de campos posibles. El formulario deberá contar además con dos botones para pasar bien al formulario anterior o bien al siguiente. El dar al usuario la posibilidad de volver al primer formulario de definición de atributos se debe a que puede que en un momento dado quiera añadir o eliminar alguno de los Attributes guardados y no lo haya detectado con anterioridad.

El tercer formulario tendrá una estructura idéntica al anterior, tan sólo que en este caso el usuario deberá dirimir cuáles de los parámetros dinámicos y cuáles los estáticos. Para ello, seleccionará los parámetros dinámicos y puesto que el atributo seleccionado como variable no debe mostrarse, los parámetros estáticos pueden identificarse por eliminación. Con el fin de que el usuario pueda corregir su error si detecta alguno, se debe habilitar un botón que le permite volver al formulario anterior. Así mismo, otro botón le permite acceder al siguiente formulario.

Finalmente, el cuarto formulario del módulo de definición de atributos se destinará a la asignación de un valor concreto a los parámetros de los atributos que no son ni variables ni parámetros dinámicos. La configuración más amigable para el

usuario se basa en una lista en la que aparecen todos los parámetros de atributo, junto a su tipo, el atributo al cual pertenecen, y la entidad concreta a la cual está ligado el atributo. De esta forma cada parámetro de atributo queda unívocamente identificado desde el punto de vista del usuario. Así pues, el formulario debe contar con un botón que permita al usuario, una vez seleccionado un Attribute Parameter de la lista, darle un valor. No obstante, un buen diseño debe tener en cuenta la necesidad del usuario de resolver su problema en el menor tiempo posible, por lo que a tal efecto se hace necesario contar con un segundo botón que inicialice todos los Attribute Parameters con valores estándar según su tipo, de tal forma que no sea tan tedioso definir los parámetros de atributos en una instancia de proceso de negocio.

El formulario también deberá contar con un botón que permita volver al formulario anterior y otro que dé al usuario la posibilidad de finalizar la definición de los atributos y sus parámetros para pasar al módulo de los Status (Estados), que trataremos en el próximo capítulo.

Así mismo, todos los formularios del módulo de definición de atributos contarán con un menú que permitirá cerrar el programa y guardar sus progresos en cualquier momento.

La justificación del orden de los cuatro formularios es simple. En primer lugar, necesitamos definir cuáles son los atributos antes de poder clasificarlos desde el punto de vista de su naturaleza como variables o parámetros. Posteriormente, será necesario separar los unos de los otros, algo que se consigue con el segundo formulario. Posteriormente, una vez tenemos claro cuáles son los parámetros, podemos clasificarlos, objetivo del tercer formulario. Por último, cuando se ha decidido cuáles de los atributos son parámetros estáticos, estamos en condiciones de asignarle un valor a sus parámetros, puesto que los parámetros de los atributos que se clasifican como variable o como parámetro dinámico no toman un valor en el nivel de instancia de proceso de negocio, sino en el nivel de ejecución, tal y como se expondrá en capítulos posteriores.

4.4 IMPLEMENTACIÓN DE LA APLICACIÓN

En esta sección se describe la implementación de la aplicación cuyo diseño se ha desarrollado anteriormente. Se trata de mostrar la apariencia de los formularios de la aplicación ya implementada, así como las funcionalidades asociadas a los mismos.

4.4.1 FORMULARIO DE DEFINICIÓN DE ATTRIBUTES

Se trata de la primera ventana que aparece cuando se comienza a configurar una instancia de proceso con el software RM Modeler (figura 4-6). Su función es la de dar un soporte al usuario para la definición de los atributos que desea incluir en la nueva instancia.

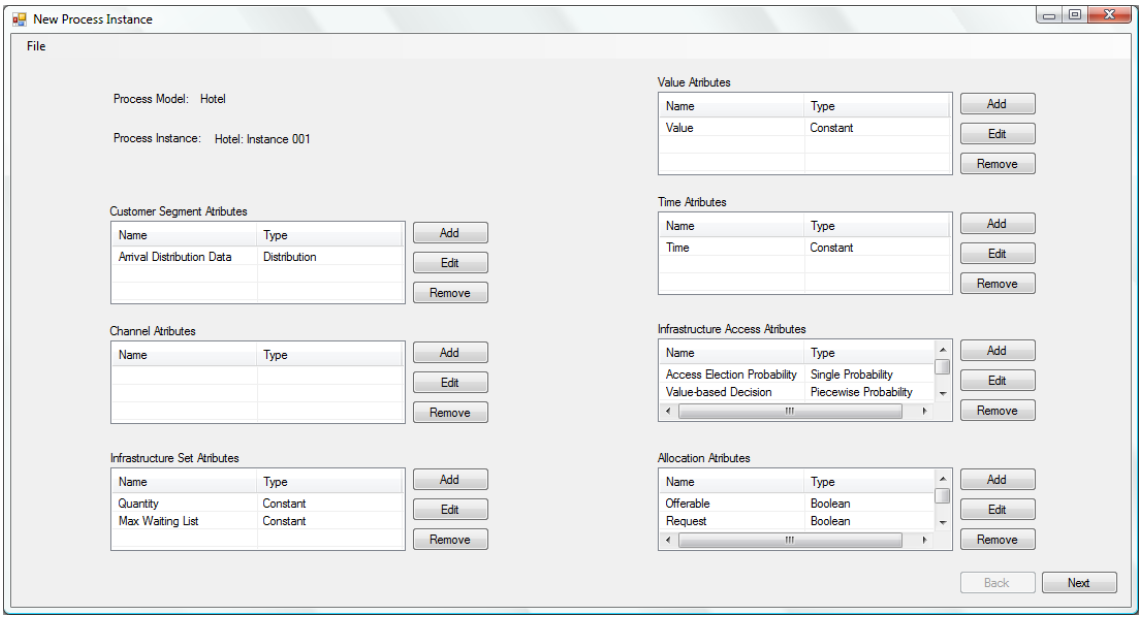


Figura 4-6: Primer formulario del módulo de definición de atributos

En la parte superior izquierda del formulario, se recogen el nombre del modelo y de la instancia de proceso, con el fin de que el usuario pueda identificar fácilmente a quién pertenecen las entidades cuyos atributos está modificando.

Cada una de las siete lista de atributos se corresponde con las siete Instance Entities, tal y como se ha explicado en el apartado de diseño. Cuentan con tres botones, denominados Add (Añadir), Edit (Editar) y Remover (Eliminar):

- Add: Permite al usuario añadir un atributo a la lista. Se hace por medio de un cuadro de diálogo que surge cuando el usuario pulsa este botón. Permite introducir el nombre del nuevo atributo a través de un text box (cuadro de texto) y elegir el tipo de atributo del cual se trata, con la asistencia de un combo box que presenta las opciones que se muestran en la figura 4-7.
- Edit: Da la posibilidad al usuario de editar el atributo de la lista que ha seleccionado antes de hacer click sobre el botón. La modificación se realiza con el mismo cuadro de diálogo que se encarga de gestionar la adición de atributos a la instancia de proceso.
- Remove: Elimina el atributo de la lista que se ha seleccionado antes de pulsar el botón.

Los nombres de los atributos que se introducen en la adición o en la edición de los mismos, deben cumplir las siguientes condiciones:

- Deben tener más de un carácter.
- Su primer carácter no puede ser un espacio.

- No pueden coincidir con el nombre de un atributo de cualquier tipo ya definido en ese mismo Process Instance. Por ejemplo, en dos Process Instance distintos podemos disponer de dos Attributes con el mismo nombre, pero no en un mismo Process Instance.

Con el fin de facilitar la configuración de una instancia de proceso por parte de los usuarios y teniendo en cuenta qué atributos serían útiles para operar algunos ejemplos en un hipotético simulador que asistiese en la resolución que nuestro software permite plantear, se han precargado los atributos que se detallan a continuación, en la tabla 4-5:

Nombre	Tipo	Entidad asociada
Arrival Distribution Data	Distribution	Customer Segment
Quantity	Constant	Infrastructure Set
Max Waiting List	Constant	Infrastructure Set
Value	Constant	Value
Time	Constant	Time
Access Election Probability	Single Probability	Infrastructure Access
Value based Decision	Piecewise Probability	Infrastructure Access
Failed Access Probability	Single Probability	Infrastructure Access
Service Time	Distribution	Infrastructure Access
Cancelled Allocation Probability	Single Probability	Infrastructure Access
Failed Allocation Probability	Single Probability	Infrastructure Access
Time.Allocated.InService	Constant	Infrastructure Access
Cancelled WL Probability	Single Probability	Infrastructure Access
Failed WL Probability	Single Probability	Infrastructure Access
Service Interrupted Probability	Single Probability	Infrastructure Access
Service Failed Probability	Single Probability	Infrastructure Access
Interruption Failed Probability	Single Probability	Infrastructure Access
Interruption Time	Constant	Infrastructure Access
Time.Served.Closed	Constant	Infrastructure Access
Time.Cancelled.Closed	Constant	Infrastructure Access
Time.Failed.Closed	Constant	Infrastructure Access
Offerable	Boolean	Allocation
Request	Boolean	Allocation
Offerable Allocation	Boolean	Allocation
Offerable WL	Boolean	Allocation

Tabla 4-5. Atributos preconfigurados en el primer formulario del módulo de definición de atributos

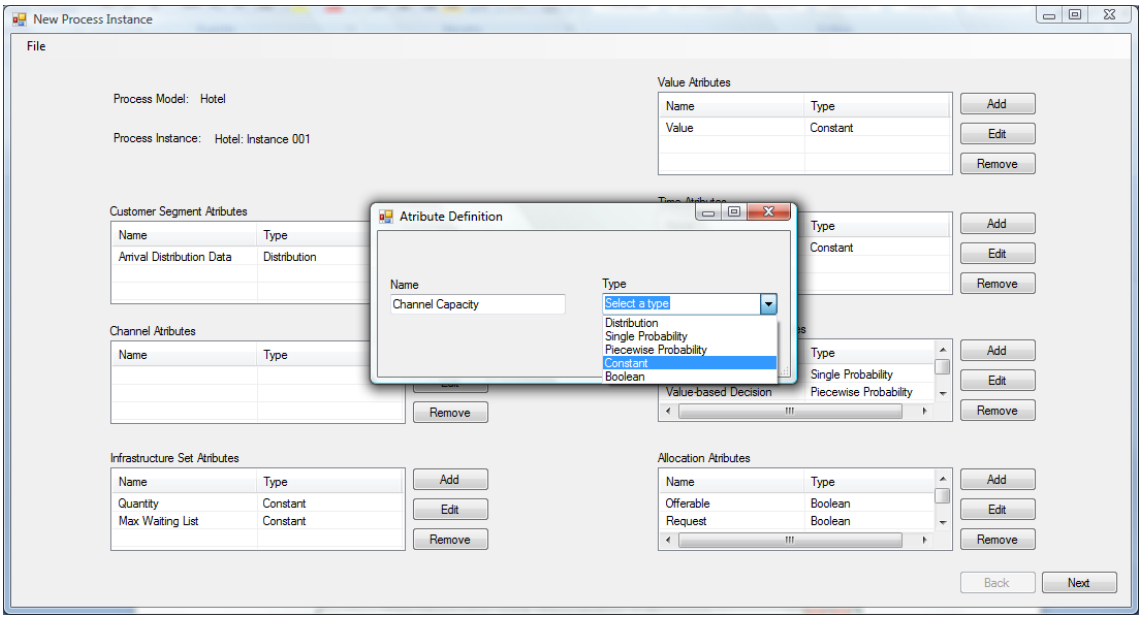


Figura 4-7. Primer formulario del módulo de definición de atributos y cuadro de diálogo de adición y edición

4.4.2 FORMULARIO DE ELECCIÓN DE VARIABLE

El elemento central del formulario (figura 4-8) es una lista en la que el software muestra todos los atributos que son de los tipos Constant y Boolean, ya que son los únicos que pueden actuar como variables. Junto al nombre de cada atributo aparecen otras características que precisan a qué entidad concreta de la instancia de proceso pertenece el atributo. Cada elemento de la lista dispone de una caja para ser marcado como variable.

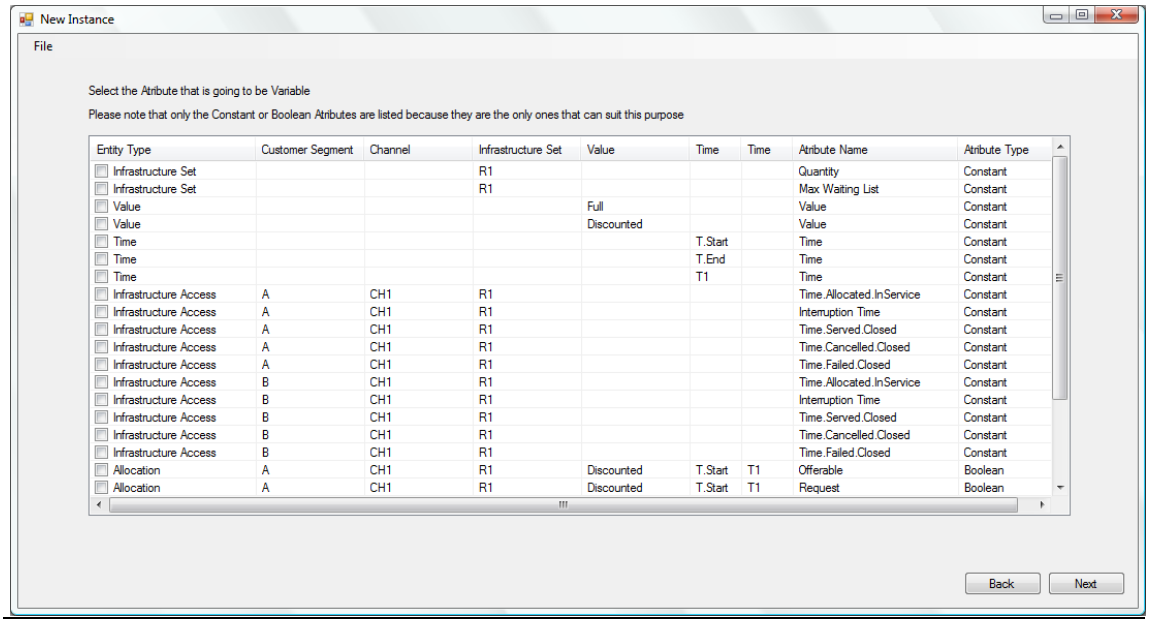


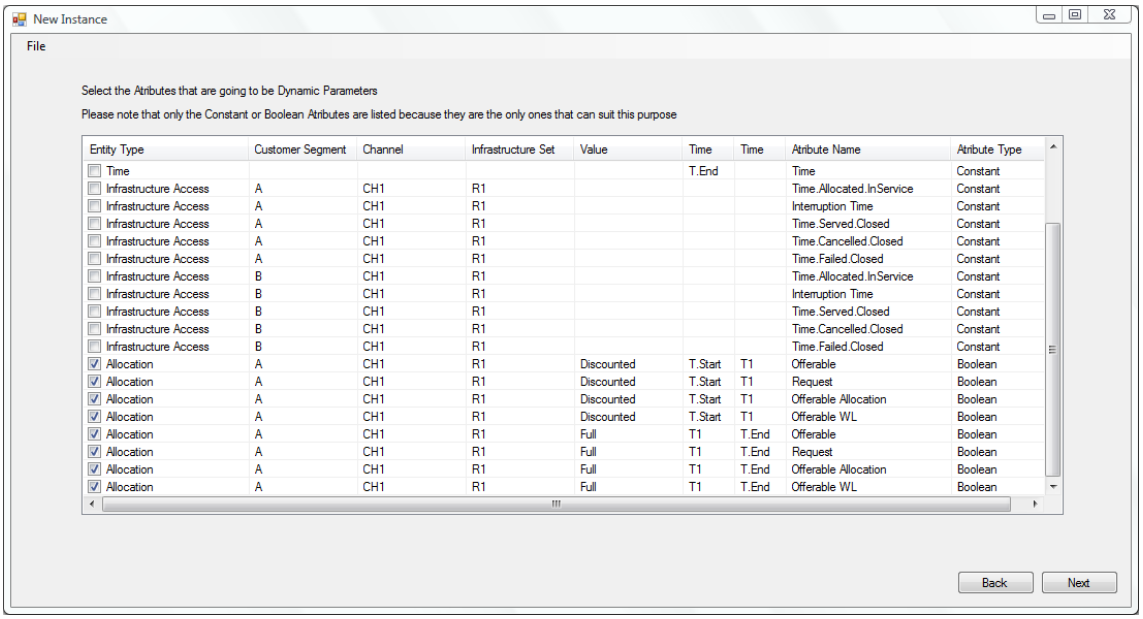
Figura 4-8. Formulario de elección de Variable del módulo de definición de atributos

En la parte inferior derecha de la pantalla se dispone de un botón denominado Next, que ejecuta un código que permite al usuario pasar al segundo formulario del módulo de definición de atributos.

El botón Back permite volver al formulario anterior, aunque se ha programado de tal forma que cuando el usuario pasa de nuevo al formulario de elección de variables, su elección previa seguirá intacta si no ha eliminado el atributo correspondiente o modificado su tipo de forma que lo invalide para actuar como variable (cambiando de Boolean o Constant a Single Probability, Piecewise Probability o Distribution).

4.4.3 FORMULARIO DE ELECCIÓN DE DYNAMIC PARAMETER

El formulario número tres del módulo de definición de atributos (figura 4-9) da soporte a la selección de los Dynamic Parameters por parte del usuario.



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Atribute Name	Atribute Type
Time					T.End		Time	Constant
Infrastructure Access	A	CH1	R1				Time.Allocated.InService	Constant
Infrastructure Access	A	CH1	R1				Interruption Time	Constant
Infrastructure Access	A	CH1	R1				Time.Served.Closed	Constant
Infrastructure Access	A	CH1	R1				Time.Cancelled.Closed	Constant
Infrastructure Access	A	CH1	R1				Time.Failed.Closed	Constant
Infrastructure Access	B	CH1	R1				Time.Allocated.InService	Constant
Infrastructure Access	B	CH1	R1				Interruption Time	Constant
Infrastructure Access	B	CH1	R1				Time.Served.Closed	Constant
Infrastructure Access	B	CH1	R1				Time.Cancelled.Closed	Constant
Infrastructure Access	B	CH1	R1				Time.Failed.Closed	Constant
Allocation	A	CH1	R1	Discounted	T.Start	T1	Offerable	Boolean
Allocation	A	CH1	R1	Discounted	T.Start	T1	Request	Boolean
Allocation	A	CH1	R1	Discounted	T.Start	T1	Offerable Allocation	Boolean
Allocation	A	CH1	R1	Discounted	T.Start	T1	Offerable WL	Boolean
Allocation	A	CH1	R1	Full	T1	T.End	Offerable	Boolean
Allocation	A	CH1	R1	Full	T1	T.End	Request	Boolean
Allocation	A	CH1	R1	Full	T1	T.End	Offerable Allocation	Boolean
Allocation	A	CH1	R1	Full	T1	T.End	Offerable WL	Boolean

Figura 4-9. Formulario de elección de parámetros dinámicos del módulo de definición de atributos

La estructura es muy parecida a la del formulario anterior. En cuanto al funcionamiento, hay dos diferencias fundamentales. Por un lado, ya no se muestran todos los atributos de los tipos Constant y Boolean, puesto que el atributo que ha sido calificado como Variable en el formulario anterior no aparece en la lista actual. Por otra parte, se pueden calificar como Dynamic Parameters tantos atributos como se desee. En el caso de que el modelo se haya definido anteriormente y se esté accediendo a una versión guardada del mismo, aparecerán como marcadas las casillas que lo estuviesen

cuando se guardó el mismo, ya que al acceder a este formulario se realiza una carga desde la base de datos.

El botón denominado Back, existente en la esquina inferior derecha del formulario, nos permite volver al segundo formulario de definición de atributos y realizar los cambios oportunos. Estas modificaciones afectan al formulario actual, añadiendo, cambiando, o eliminando los atributos de la lista. El botón Next, que se halla junto al botón anterior, permite al usuario acceder al siguiente formulario una vez ha plasmado sus preferencias. El guardado de la información en la base de datos se realiza cuando se pulsa cualquiera de estos dos botones.

4.4.4 FORMULARIO DE EVALUACIÓN DE LOS ATTRIBUTE PARAMETERS

El cuarto formulario del módulo de Attributes otorga al usuario la facultad de dar un valor a los Attribute Parameters que tendrá cabida en la instancia de proceso.

Una lista que ocupa la práctica totalidad del formulario, tal y como se aprecia en la figura 4-10, muestra los Attribute Parameters de los atributos clasificados como Static Parameters. Dos botones permiten modificar el valor de los mismos. El primero da la posibilidad de editar el valor de un Attribute Parameter concreto. Se consigue por medio de un cuadro de diálogo, como el que aparece en la figura 4-11, que solicita el nuevo valor y que además es sensible al tipo de datos que se introducen. Por ejemplo, si se nos pide un número, el software no aceptará que el usuario le asigne al parámetro un texto como valor. El siguiente código es el que se asegura de la calidad de los datos introducidos en ese cuadro de diálogo:

```
'Si el parámetro de atributo es el tipo de distribución
If lblAttributeParameter.Text = "Distribution Type" Then
      'Se asegura que se ha optado por alguna de las distribuciones que
      ofrece el programa
    If cbxAP.SelectedItem <> "" Then

      frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
      tem(10).Text = cbxAP.SelectedItem
    End If
    'Si el parámetro de atributo es un parámetro de una distribución
ElseIf lblAttributeParameter.Text = "Distribution Parameter 1" Or _
    lblAttributeParameter.Text = "Distribution Parameter 2" Then
    'Se asegura de que el dato introducido es de tipo numérico y además
            es un número mayor que cero
    If IsNumeric(tbxAP.Text) And Val(tbxAP.Text) > 0 Then

      frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
      tem(10).Text = tbxAP.Text
    Else
        MessageBox.Show("Must be a number greater than 0")
    End If
    'Si el parámetro de atributo es un valor de una probabilidad
ElseIf lblAttributeParameter.Text = "Probability Value" Then
```

```
            'Se asegura de que el dato introducido es de tipo numérico y además
             es un número mayor o igual que cero y menor o igual que uno
            If IsNumeric(tbxAP.Text) And Val(tbxAP.Text) >= 0 And Val(tbxAP.Text)
            <= 1 Then
                Probability = tbxAP.Text

             frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
             tem(10).Text = Format(Probability, "0.0000")
            Else
                MessageBox.Show("Must be a number between 0 and 1")
            End If
             'Si el parámetro de atributo es el número de intervalos de una
             probabilidad definida a trozos
        ElseIf lblAttributeParameter.Text = "Number of Intervals" Then
             'Se asegura de que el dato introducido es de tipo numérico y además
             es un número mayor o igual que uno y menor o igual que cinco, el
             máximo permitido
            If IsNumeric(cbxAP.SelectedItem) And Val(cbxAP.SelectedItem) >= 1 And
            Val(cbxAP.SelectedItem) <= 5 Then

             frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
             tem(10).Text = cbxAP.SelectedItem
            Else
                MessageBox.Show("Must be an integer number between 1 and 5")
            End If
            'Si el parámetro de atributo es un valor mínimo, máximo o constante
        ElseIf lblAttributeParameter.Text = "Min Value" Or _
        lblAttributeParameter.Text = "Max Value" Or _
        lblAttributeParameter.Text = "Constant Value" Then
            'Se asegura de que el dato introducido es de tipo numérico
            If IsNumeric(tbxAP.Text) Then

             frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
             tem(10).Text = tbxAP.Text
            Else
                MessageBox.Show("Must be a number")
            End If
            'Si el parámetro de atributo es valor booleano
        ElseIf lblAttributeParameter.Text = "Boolean Value" Then
             'Se asegura de que se ha elegido el valor true o false del menú
             desplegable
            If cbxAP.SelectedItem <> "" Then

             frm09NewInstance.lvwAttributes.Items.Item(IndexSelected).SubItems.I
             tem(10).Text = cbxAP.SelectedItem
            End If
        End If
```

El segundo botón tiene como función otorgar a todos los parámetros de atributo un valor por defecto en función de su tipo y, en algunos casos, del nombre del atributo. El objetivo es que sea rápido para el usuario definir una instancia con valores estándar y que posteriormente, sirviéndose del botón de edición individual que hemos descrito con anterioridad, pueda realizar todos los ajustes que considere necesarios. Tiene la ventaja de resultar muy útil para acelerar el desarrollo de una instancia de proceso a la vez que se mantiene la flexibilidad del software.

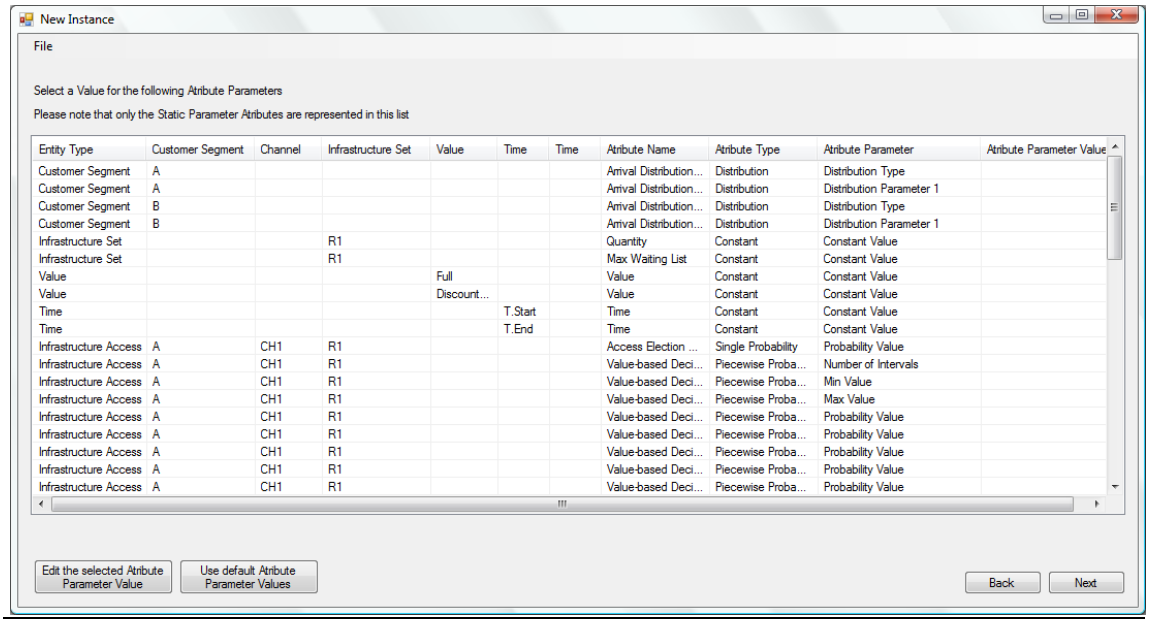


Figura 4-10. Formulario de evaluación de Attribute Parameters del módulo de definición de atributos

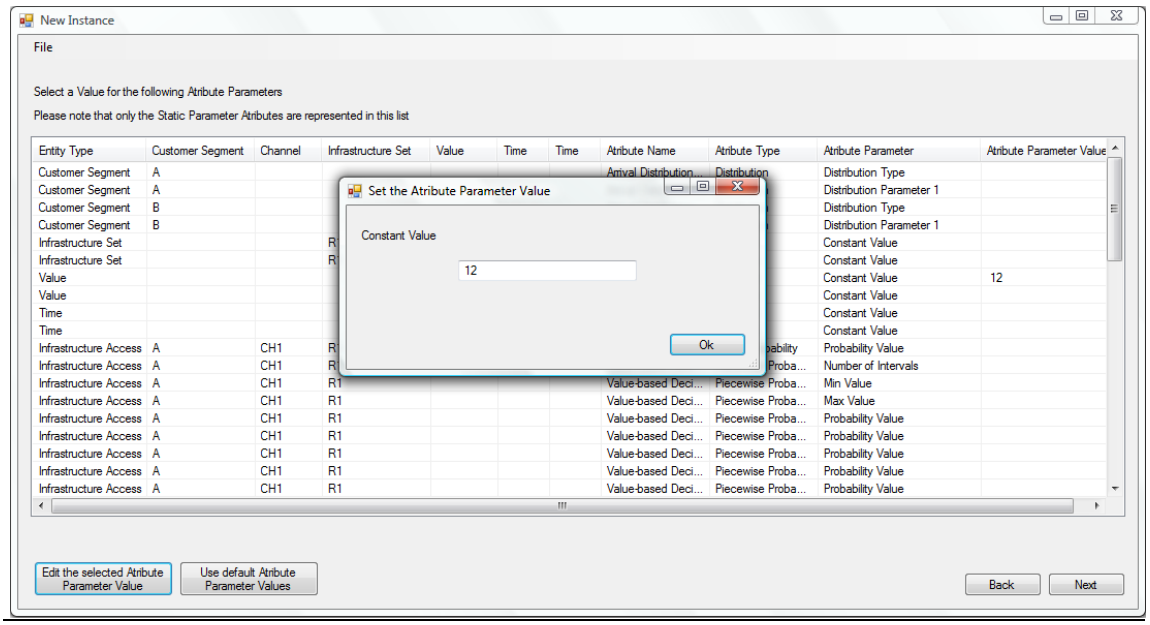


Figura 4-11. Cuadro de diálogo de evaluación de un Attribute Parameter del tipo Constant Value

Este es fundamentalmente el fragmento de código que tiene como misión configurar los parámetros de atributo con los valores predefinidos:

```
'Inicializamos los valores de los Attribute Parameters
'Bucle que recorre todos los Attribute Parameters
For i = 0 To lvwAttributes.Items.Count - 1
    'Si se trata de un Constant Value
```

```
 If lvwAttributes.Items.Item(i).SubItems.Item(9).Text = "Constant
 Value" Then
    'Valor 0, al menos en un principio, pues puede modificarse
    después si concurren otras circunstancias
    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "0"
End If
'Si se trata de un tipo de distribución
 If lvwAttributes.Items.Item(i).SubItems.Item(9).Text =
 "Distribution Type" Then
    'Por defecto es exponencial
    lvwAttributes.Items.Item(i).SubItems.Item(10).Text ="Exponential"
    'Parámetro de distribución
 ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Tex=
 "Distribution Parameter 1" Then
    'Valor 10
    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "10"
 ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text =
 "Probability Value" Then
    If lvwAttributes.Items.Item(i).SubItems.Item(8).Text = "Piecewise
    Probability" Then
        'Si el nombre del atributo es Access Election Probability
   ElseIf lvwAttributes.Items.Item(i).SubItems.Item(7).Text =
 "Access Election Probability" Then
        CSName = lvwAttributes.Items.Item(i).SubItems.Item(1).Text
        NumberCSName = 0
        For j = 0 To lvwAttributes.Items.Count - 1
               If lvwAttributes.Items.Item(j).SubItems.Item(1).Text =
               CSName And
               lvwAttributes.Items.Item(j).SubItems.Item(7).Text =
               "Access Election Probability" Then
                NumberCSName = NumberCSName + 1
               End If
        Next
        'Reparte la probabilidad a partes iguales entre todos los
        Infrastructure Access con el mismo Customer Segment
        Probability = 1 / NumberCSName

        lvwAttributes.Items.Item(i).SubItems.Item(10).Text =
        Format(Probability, "0.0000")
    Else
        lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "0.0001"
    End If
 ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text = "Number
 of Intervals" Then
    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "5"
        'Se inicializa el valor de los valores de probabilidad de una
        probabilidad definida a trozos
    If lvwAttributes.Items.Item(i + 3).SubItems.Item(9).Text =
    "Probability Value" And _
    lvwAttributes.Items.Item(i + 4).SubItems.Item(9).Text =
    "Probability Value" And _
    lvwAttributes.Items.Item(i + 5).SubItems.Item(9).Text =
    "Probability Value" And _
    lvwAttributes.Items.Item(i + 6).SubItems.Item(9).Text =
    "Probability Value" And _
    lvwAttributes.Items.Item(i + 7).SubItems.Item(9).Text =
    "Probability Value" Then

    lvwAttributes.Items.Item(i + 3).SubItems.Item(10).Text = "0.9000"
    lvwAttributes.Items.Item(i + 4).SubItems.Item(10).Text = "0.8000"
    lvwAttributes.Items.Item(i + 5).SubItems.Item(10).Text = "0.6500"
    lvwAttributes.Items.Item(i + 6).SubItems.Item(10).Text = "0.4500"
```

```
            lvwAttributes.Items.Item(i + 7).SubItems.Item(10).Text = "0.2000"
            End If
        ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text = "Min
        Value" Then
            lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "0"
        ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text = "Max
        Value" Then
            lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "100"
        ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text =
        "Constant Value" Then
            'Tiempos
          If lvwAttributes.Items.Item(i).SubItems.Item(0).Text = "Time" Then
                If lvwAttributes.Items.Item(i).SubItems.Item(5).Text =
                "T.Start" Then
                    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "0"
                ElseIf lvwAttributes.Items.Item(i).SubItems.Item(5).Text =
                "T.End" Then
                   lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "100"
                Else
                    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "50"
                End If
                'Valor
            ElseIf lvwAttributes.Items.Item(i).SubItems.Item(0).Text
        ="Value"                                          Then
                lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "50"
                'Atributos de Infrastructure Set
            ElseIf lvwAttributes.Items.Item(i).SubItems.Item(0).Text =
        "Infrastructure Set" Then
                'Por nombre:
                If lvwAttributes.Items.Item(i).SubItems.Item(7).Text =
                "Quantity" Then
                    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "5"
                ElseIf lvwAttributes.Items.Item(i).SubItems.Item(7).Text =
                "Max Waiting List" Then
                    lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "0"
                End If
            End If
            'Todo valor lógico por defecto lo consideramos verdadero
        ElseIf lvwAttributes.Items.Item(i).SubItems.Item(9).Text = "Logical
Value" Then
            lvwAttributes.Items.Item(i).SubItems.Item(10).Text = "True"
      End If
```

5

MÓDULO DE DEFINICIÓN DE ESTADOS

Con idéntica estructura que los dos capítulos anteriores, el diseño conceptual del modelo de estados ocupa la primera sección del presente capítulo. Posteriormente, se expone el modelo de datos del módulo, con un análisis detallado del diseño de la base de datos que procura soporte a la aplicación, incluyendo un diagrama explicativo de la misma. La tercera sección está dedicada al diseño de la aplicación, explicando su estructura con la asistencia de un diagrama de estados UML. Finalmente, la cuarta sección recoge el desarrollo de la implementación del módulo del software asociado a la definición de estados de los accesos a infraestructuras (Infrastructure Access) y asignaciones (Allocations).

5.1 DISEÑO CONCEPTUAL

Este tercer módulo comprende la definición de los estados ligados a la instanciación de un proceso de negocio. Los Status (Estados) son las distintas fases en la cuales puede encontrarse un Access o una Allocation durante la ejecución de un proceso de negocio. Por ejemplo, un acceso de un cliente puede que tenga que pasar por varios estados previos antes de llegar a convertirse en una asignación de infraestructuras. En el caso de un cliente de un hotel que intenta reservar una habitación doble a través de Internet para un fin de semana, en primer lugar deberá acceder a la página e indicar sus preferencias, posteriormente sabrá si hay habitaciones dobles disponibles en la fecha deseada, con lo que su acceso será aceptado o rechazado. Una vez se decide a solicitar la reserva, tendrá que pasar por el proceso de pago, donde puede que consiga la reserva definitiva de la habitación o bien nunca llegue a producirse por algún tipo de circunstancia, como puede ser un fallo debido a los sistemas de información del medio de pago.

Como se aprecia en la figura 5-1, el modelo de Estados, que también puede denominarse como modelo de la dinámica de una instancia de un proceso (Process Instance Dynamics), pretende dar una respuesta conceptual a la necesidad de incluir en la función objetivo (que también se encuentra en el nivel de Process Instance), términos que van ligados a la evolución del sistema durante la ejecución del proceso.

Así pues, un Infrastructure Access está ligado a uno o más Status (Estados). Así mismo, varios Infrastructure Access pueden compartir un mismo Status. Los Status pueden clasificarse como Status propios de los accesos (Access Status) o como Status propios de las asignaciones (Allocation Status). Esta clasificación es disjunta y completa. Es decir, un Status o estado sólo puede formar parte de uno de los dos subconjuntos y además obligatoriamente debe pertenecer a uno de los dos. Todo acceso o asignación que deriva del Infrastructure Access asociado a un estado, puede llegar a encontrarse en ese estado. Además, cada Status Allocation se relaciona con varios Values, y cada Value puede asociarse a más de un Status Allocation.

Para alcanzar un Status determinado, los accesos y las asignaciones sufren cambios de estado, denominados en el diagrama de clases UML como Status Change. En cada uno de estos Status Change, un Status tiene el rol de Status previo (Previous

Status) y otro el de Status siguiente (Next Status). Además un Status puede dar lugar a varios Status distintos, esto es, desempeñar el papel de Previous más de una vez; y de forma análoga un mismo Status puede ser el estado Next en más de un Status Change.

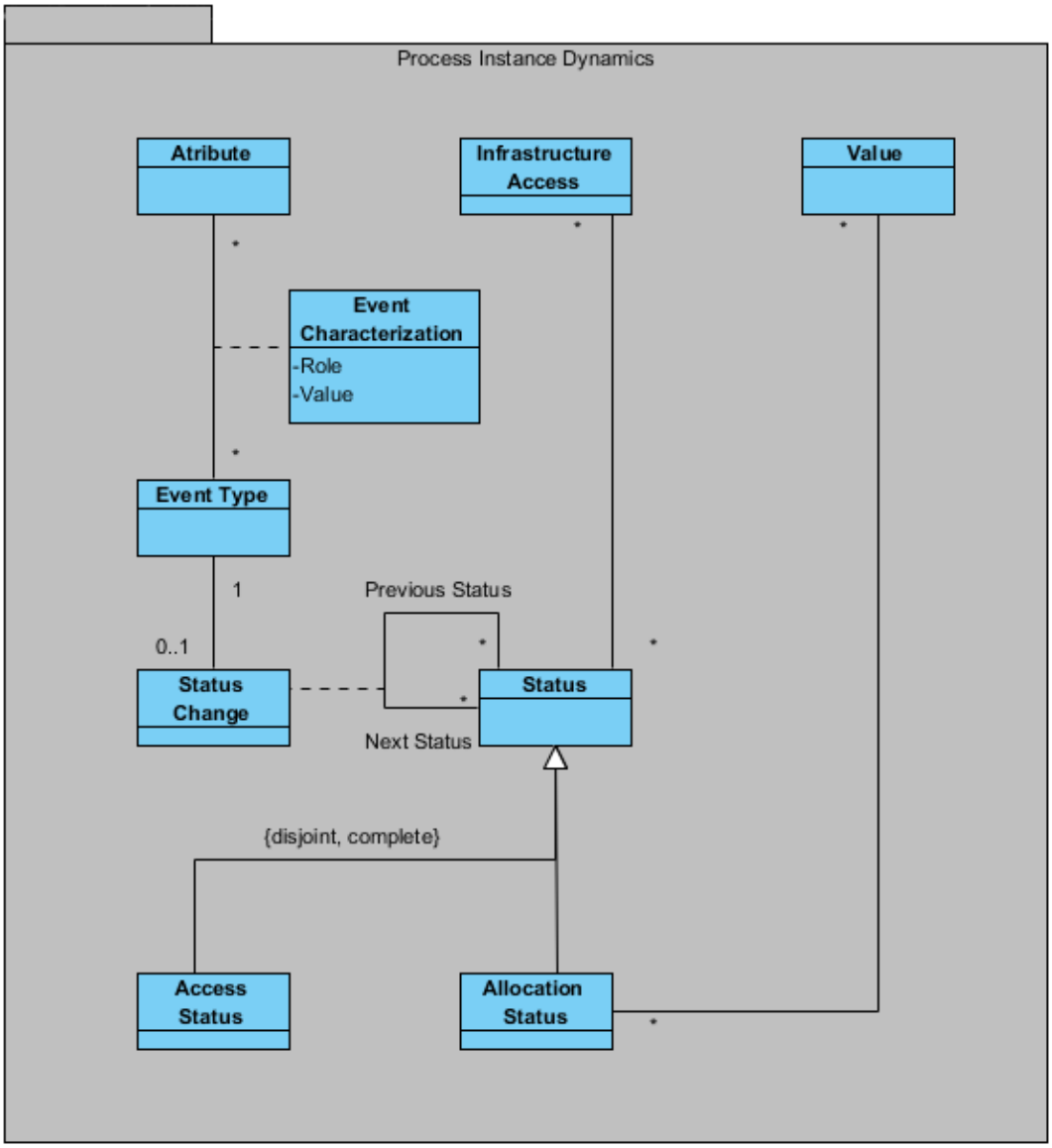


Figura 5-1. Diagrama UML de Process Instance Dynamics o modelo de Status

Por otra parte, algunos atributos, al tomar un determinado valor, y en ocasiones en colaboración con otros atributos, determinan la ocurrencia de un Event Type (tipo de evento). Los Event Types son las categorías de los sucesos que pueden darse durante la ejecución de un proceso de negocios. La relación entre el atributo y el tipo de evento, que hemos explicado anteriormente, se conoce como Event Characterization (caracterización del evento). No existe ningún tipo de restricción que impida que un mismo atributo, con un valor y un rol distinto o no, pueda dar lugar a más de un tipo de evento. En el sentido contrario, un mismo tipo de evento puede estar relacionado con

más de un atributo. Por ejemplo, un tipo de evento que requiera que dos o más atributos tomen un determinado valor a la vez para que ocurra un evento de ese tipo.

Finalmente, dado que hemos definido tanto un Event Type como un Status Change, podemos cerrar el círculo del modelo de la dinámica de la instancia de un proceso. Un Event Type puede provocar un Status Change, si bien no todos los Event Types se relacionan con un Status Change, como es el caso de los tipo de eventos relacionados con un Time concreto, cuya función habitual es influir sobre la vigencia de las distintas Allocations.

Recuperaremos el ejemplo del establecimiento de alquiler de automóviles para explicar mejor el Process Instance Dynamics. Empezando con los Status de los accesos, se presentan en la tabla 5-1:

Previous Status	Next Status
	Customer Access
Customer Access	Accepted Access
Customer Access	Rejected Access
Accepted Access	Requested Allocation Access
Accepted Access	Rejected Allocation Access
Requested Allocation Access	Failed Access
Requested Allocation Access	Successful Access

Tabla 5-1. Status Changes de ejemplo

Interpretando la información que aparece en la tabla, un acceso de un cliente al sistema de reservas de la compañía de alquiler de coches puede pasar por varios Status. En primer lugar, se crea el acceso, cuyo estado inicial es Customer Access (Acceso de Cliente). Ese estado no deriva de ningún otro estado, que es lo que explica el espacio en blanco que aparece en la columna Previous Status del primer Status Change. Una vez el cliente ha accedido al sistema, pueden ocurrir dos cosas:

- Puede que la demanda del cliente sea atendida porque existe una oferta de asignación disponible para él. La consecuencia es un cambio de Status de Customer Access a Accepted Access (Acceso Aceptado).
- La otra posibilidad es que no exista tal oferta, bien porque nunca ha existido, bien porque no está vigente en ese momento o bien porque está vigente pero toda la oferta está cubierta con peticiones anteriores de clientes que tuvieron mayor capacidad de anticipación. La consecuencia es un cambio de Status del acceso de Customer Access a Rejected Access (Acceso Rechazado).

Suponiendo que un cliente mayor de 25 años desea alquilar un coche compacto, a través del único canal disponible (el sitio web de la empresa), y que no todos los

coches de ese tipo están ya reservados, su acceso es aceptado. A continuación, se le mostrará vía web el precio de realizar la reserva (Value del Allocation correspondiente). En función del valor que le aporta recibir el servicio y del coste del mismo, el cliente tomará una decisión sobre la posibilidad de alquilar o no el coche. Si decide alquilarlo, el Status del acceso pasará de Accepted Access a Requested Allocation Access (Acceso de Asignación Solicitada). Si la decisión es la contraria, el acceso pasará de un Status de Accepted Access a otro de Rejected Allocation Access (Acceso de Asignación Rechazada).

En caso de que el cliente optase por la opción de contratar el alquiler del coche, aún tendría que realizar el pago. Para simular la posible aparición de problemas técnicos u otro tipo de imprevistos durante la formalización de la transacción, pueden incluirse dos nuevos Status Changes:

- De Requested Allocation Access a Successful Access (Acceso Exitoso).
- De Requested Allocation Access a Failed Access (Acceso Fallido).

Dependiendo de una determinada probabilidad, a priori bastante baja, habrá ciertas transacciones que nunca llegarán a cerrarse (Failed Access), si bien en la mayor parte de los casos una petición de asignación terminará con un acceso exitoso y la correspondiente reserva.

Si bien lo hemos mencionado de manera abstracta, cabría conocer exactamente cuáles podrían ser los Event Types que disparasen los cambios de estado de un acceso en nuestro ejemplo. Previamente, debemos profundizar en los Event Types y su posible clasificación (figura 5-2).

En un primer nivel de clasificación, podemos distinguir tres tipos de Event Types:

- Boolean Event Types (Tipos de Eventos Booleanos): Son tipos de eventos que suceden cuando un determinado atributo de tipo booleano toma un valor lógico verdadero o falso.
- Probability Event Types (Tipos de Eventos de Probabilidad): Son tipos de eventos ligados a alguna forma de cálculo de probabilidades.
- Time Event Types (Tipos de Eventos de Tiempo): Estos tipos de eventos aparecen en relación con un momento de tiempo determinado.

La clasificación es del tipo overlapping (superpuesta) y completa, es decir, un Event Type puede pertenecer a más de una de las categorías anteriores, pero en cualquier caso sólo a alguna o algunas de ellas. Siguiendo con la descripción de la taxonomía de los Event Types, los Probability Event Types se subdividen de forma disjunta y completa en:

- Complementary Probabilities Event Types (Tipos de Eventos de Probabilidades Complementarias): Un tipo de evento probabilísticos que

tiene la particularidad de que la suma de los valores de probabilidad de los atributos relacionados con el mismo debe ser igual a 1.

- Single Probability Event Types (Tipos de Eventos de Probabilidad Simples): Se dan cuando tras realizar el cálculo de probabilidad sobre el valor de probabilidad del atributo al cual está ligado ese tipo de evento alcanza un valor verdadero o falso. Por ejemplo, si el valor de probabilidad es 0.6, habrá un 60% de posibilidades de que el resultado del cálculo probabilístico sea verdadero. En caso de serlo, si el valor suscrito en el Event Characterization era verdadero, el Event Type queda validado.
- Piecewise Probability Event Types (Tipos de Eventos de Probabilidad Definida a Trozos): Se relacionan con dos atributos, uno del tipo Piecewise Probability y otro del tipo Constant. Este último determina qué trozo de la función de probabilidad se evalúa. A partir de ahí, su funcionamiento es idéntico al de los Single Probability Event Types.

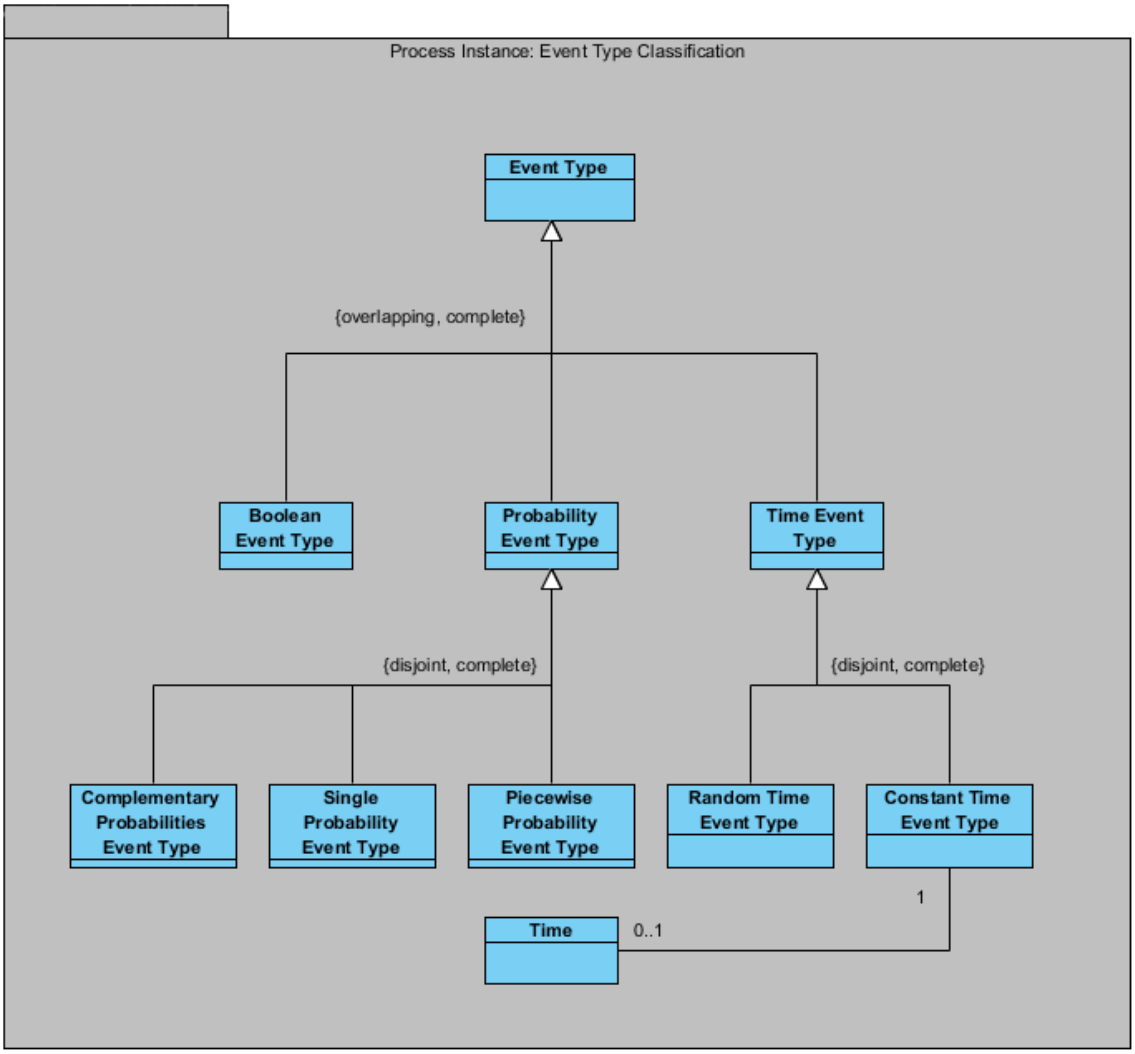


Figura 5-2. Diagrama de clases UML de la clasificación de Event Type

Los Time Event Types también pueden subclasificarse en Random Time Event Types y Constant Time Event Types:

- Random Time Event Types (Tipos de Eventos de Tiempo Aleatorio): Son tipos de eventos que suceden cuando ha transcurrido un tiempo aleatorio desde que está activo el Status precedente en el Status Change en el que intervienen.
- Constant Time Event Types (Tipos de Eventos de Tiempo Constante): Pueden estar asociados a un Time o no. Si lo están, no aparecen ligados a ningún Status Change y tan sólo pueden determinar cambios en la vigencia de los Allocations. En caso de no estar relacionados con un Time, sí lo están con un Status Change y funcionan igual que los Random Time Event Type sólo que en este caso el lapso temporal es fijo, no aleatorio.

Siguiendo con el ejemplo de la compañía de alquiler de coches, cada uno de los Status Change que aparecen en la tabla 5-1 van ligados a un Event Type, que podríamos denominar mediante el nombre del Status con el rol de Next en el Status Change correspondiente. Si se hace así, los Event Types, teniendo en cuenta la clasificación que acaba de explicarse, podría caracterizarse de la siguiente forma (tabla 5-2):

Event Type	Classification1	Classification2	Attribute1	Value1	Attribute2	Value2
Customer Access	Random Time	Complementary Probabilities	Arrival Distribution Data		Access Election Probability	True
Accepted Access	Boolean		Offerable	True		
Rejected Access	Boolean		Offerable	False		
Requested Allocation	Boolean		Request	True		
Rejected Allocation	Boolean		Request	False		
Failed Access	Single Probability		Failed Access Probability	True		
Successful Access	Single Probability		Failed Access Probability	False		

Tabla 5-3. Ejemplo de caracterización de Event Types

El Event Type Customer Access es el único de los que se muestran en este ejemplo que pueden clasificarse de dos maneras distintas. Por un lado los clientes llegan en momentos de tiempo aleatorio siguiendo una determinada distribución de llegadas, y

por otro se determina el Infrastructure Access que llevarán a cabo de acuerdo a la probabilidad que se le ha otorgado por parte del creador de la instancia del proceso a cada una de las posibilidades.

Una vez el cliente accede, su acceso se acepta o se rechaza en función de si el Infrastructure Access ofrecido se oferta o no, lo cual lo determina el valor lógico del atributo Offerable. En el caso de que el valor sea verdadero, para que el acceso se convierta en una petición de reserva de alquiler de un automóvil será necesario que el cliente decida si el precio ofrecido le conviene, y hará la petición o no, con lo que el parámetro dinámico Request tomará un valor verdadero o falso, respectivamente. Si Request adquiere un valor verdadero, finalmente habrá que dirimir si el acceso falla, con una probabilidad denominada Failed Access Probability (Probabilidad de Acceso Fallido), o si por el contrario se trata de un acceso exitoso y el proceso de reserva sigue adelante.

5.2 MODELO DE DATOS

En esta sección se expone el modelo de datos de este módulo. Para ello, no serviremos del diagrama IDEF1X elaborado con ERWin que aparece en la figura 5-3, y en el cual puede apreciarse el diseño lógico del segmento de la base de datos correspondiente al modelo del Process Instance Dynamics o modelo de Status.

En lo que respecta al diseño de modelo de datos, la primera decisión importante consiste en decidir qué tablas son necesarias. Se considera cada una de las entidades que aparecen en el diagramas de clases UML representado en la figuras 5-1. Todas requieren una tabla propia en la base de datos. Así pues, las tablas nuevas que será necesario añadir a la base de datos son: TableStatus, StatusChange, EventType y EventCharacterization, tal y como puede apreciarse en la figura 5-3.

La tabla en la cual se anota el primer registro cuando se comienza a definir la dinámica de una instancia de proceso de negocio es TableStatus. Su denominación se debe a que Status es una palabra clave del lenguaje SQL. La clave primaria de esta tabla es en parte heredada de la tabla InfrastructureAccess (ID_ProcessModel e ID_ProcessInstance) y en parte propia (ID_Status). Además la tabla cuenta con un campo adicional, Name_, un atributo que se corresponde con el nombre que el usuario decide darle al Status. El guión bajo del final es necesario porque “Name” es una palabra clave del lenguaje SQL. El segundo atributo de la tabla TableStatus es Type_, de tipo texto, que permite almacena la información sobre la clasificación del Status como Access Status o como Allocation Status. La barra baja final de Type_ tiene el mismo origen que la de Name_. Las claves ajenas ID_InfrastructureAccess e ID_Value no forman parte de la clave principal y nos ayudan a identificar un Status como asociado a una familia de accesos o de asignaciones.

La clave de la tabla StatusChange es totalmente heredada. Recibe de TableStatus ID_ProcessModel e ID_ProcessInstances. También recibe ID_PreviousStatus e

ID_NextStatus de TableStatus, pero de dos registros distintos de dicha tabla, que entregan cada uno de ellos su clave ID_Status y se le cambia de nombre según el rol que desempeña cada Status en el Status Change correspondiente.

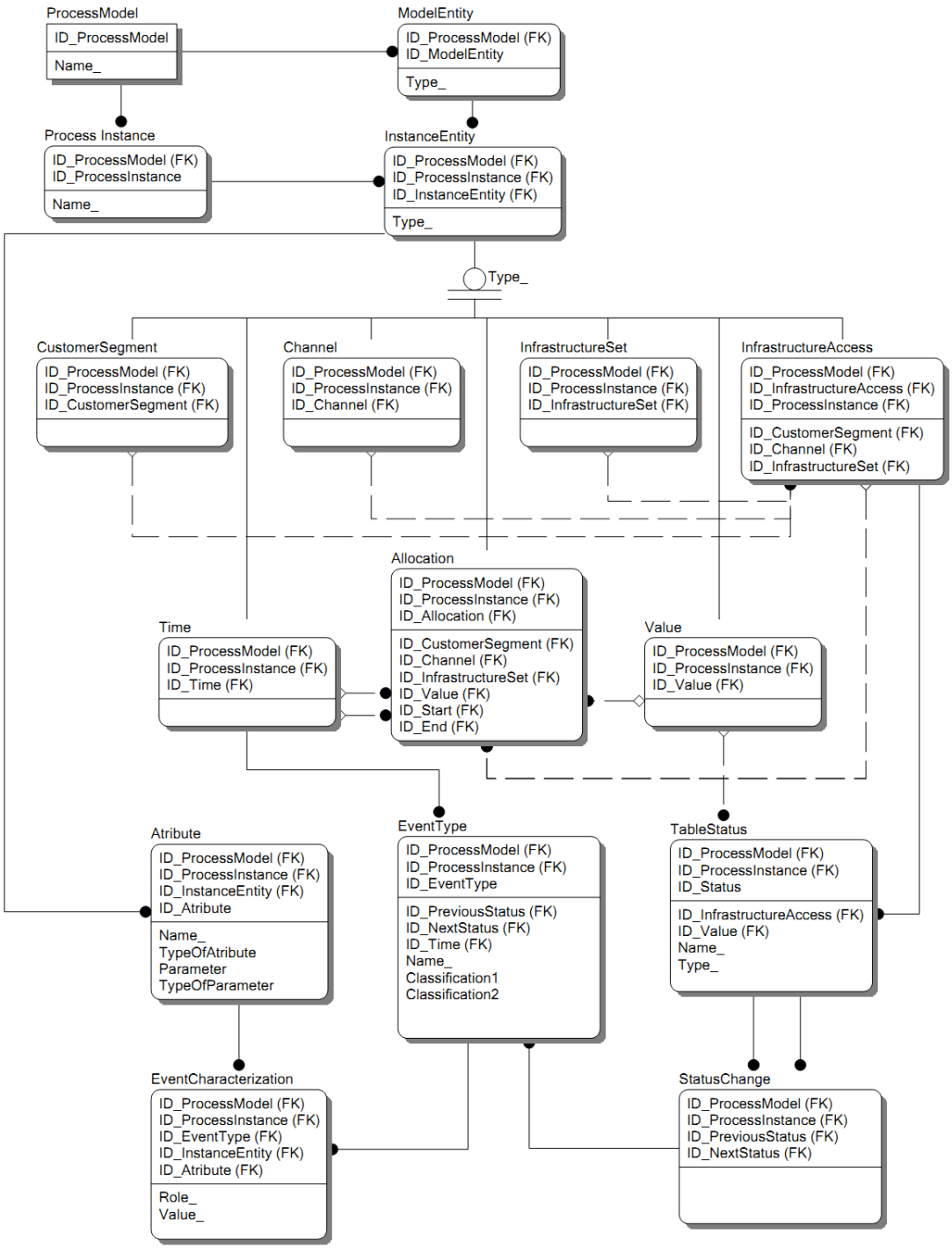


Figura 5-3. Diagrama IDEF1X del segmento de la base de datos correspondiente al modelo de Status

El siguiente paso lógico del análisis es estudiar la tabla EventType. Su clave principal está formada por las claves ajenas ID_ProcessModel e ID_ProcessInstance, que hereda bien de Time o bien de StatusChange, y la clave propia ID_EventType. Además, cada registro de la tabla EventType cuenta con varias claves ajenas que no forman parte de la clave principal. Son ID_PreviousStatus, ID_NextStatus e ID_Time. Las dos primeras almacenan información cuando el EventType está ligado a un StatusChange. La última es útil en el caso de que el tipo de evento no esté relacionado con un StatusChange sino con un Time, tal y como se explicó en el diseño conceptual del apartado 5.1.

Cada registro de EventType cuenta con un nombre que se almacena en el atributo Name_. Además, los atributos Classification1 y Classification2 nos permiten guardar la información sobre la taxonomía de cada EventType.

De la relación entre un EventType y un Attribute deriva un Event Characterization. Es por ello que la tabla EventCharacterization carece de clave propia. La clave principal está integrada en su totalidad por claves ajenas, que son todas las claves que forman la clave principal de Attribute (ID_ProcessModel, ID_ProcessInstance, ID_InstanceEntity e ID_Attribute) y además la única clave propia de EventType, ID_EventType.

5.3 DISEÑO DE LA APLICACIÓN

En esta sección se expone el proceso de diseño del módulo la aplicación que da soporte a la definición de la dinámica de una instancia de proceso. Se desarrolla cuál es la estructura y se muestra un diagrama de secuencia de UML.

La definición de los estados presentes en una instancia de proceso no requiere en absoluto que el programa haga referencia directa a de las instancias de Infrastructure Acces, al menos en el formulario correspondiente a la definición de los Status, si tomamos la decisión de que todos los Access y Allocations sigan el mismo itinerario de estados. La opción alternativa sería permitir que los Status de cada Infrastructure Access fueran distintos, pero hay dos motivos que nos inclinan a rechazar tal extremo. Por un lado, no sería coherente con el diseño de la parte de la aplicación correspondiente al módulo de atributos, pues los atributos se asignan a todas las instancias de un mismo tipo de entidades de la instancia de proceso, y si los Status, y por tanto los Event Types, son distintos en función del Infrastructure Access, el resultado sería la existencia de atributos cuya definición, a pesar de que se ha realizado, no tiene ningún sentido para ciertas instancias de alguno o algunos tipos de Instance Entities.

Por otra parte, provocaría que el proceso de definición de Status, Status Changes, Event Types y Event Characterizations fuese extremadamente complejo, con lo que el software perdería gran parte de su potencia como herramienta rápida y concisa a cambio de una flexibilidad que añade escaso valor, ya que en la gran mayoría de los

casos todos los Infrastructure Access afrontan un proceso de reservas extremadamente parecido.

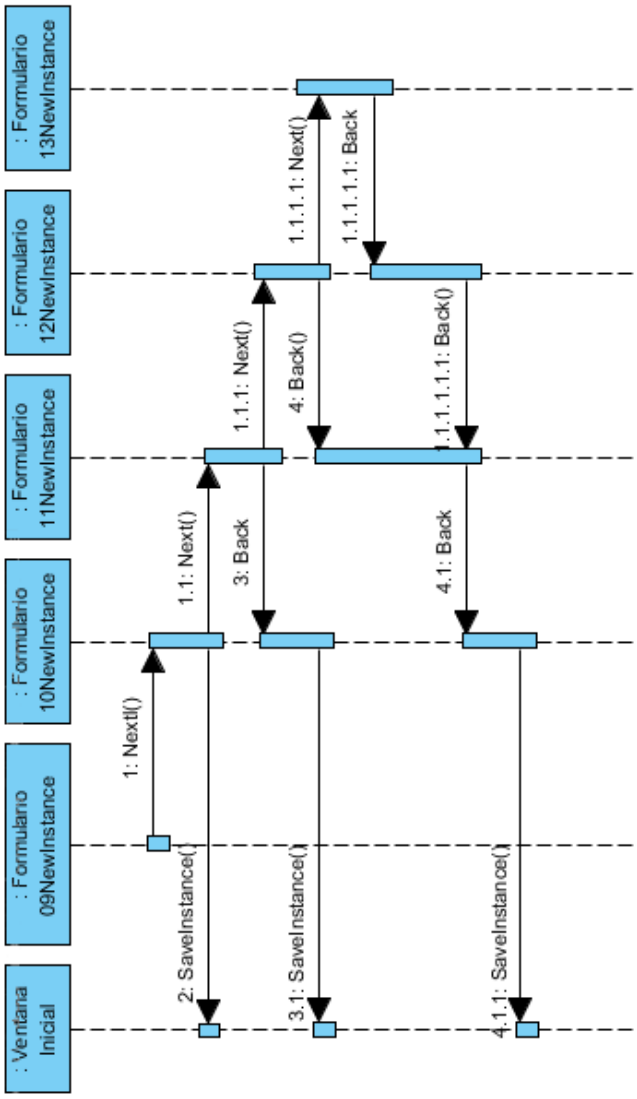


Figura 5-4. Diagrama de secuencia en UML del módulo de la aplicación asociado a la definición de Status

En consecuencia, en el primer formulario se definen los Status válidos para todos los Infrastructure Access, que serán los mismos, así como los correspondientes Status Changes, con los que ocurre algo semejante. La intención es que el formulario conste de dos listas. En la primera deben aparecer los Status, tanto los que se decidan preconfigurar como aquellos que haya añadido el usuario mediante un botón dispuesto a tal efecto. También deben habilitarse las opciones de editar y borrar Status ya añadidos. En la otra lista se enumerarán los Status Changes, junto a tres botones con las mismas funcionalidades que los de la otra lista: añadir, editar y eliminar elementos de la lista. Por último, el formulario deberá contar con un botón para avanzar al siguiente

formulario y otro para volver al formulario anterior, que se describió en el capítulo dedicado al módulo de atributos.

El segundo formulario de este módulo debe permitir al usuario configurar los Event Types que desea incluir en esta instancia del proceso. Para ello, contará con una lista en la que se mostrará el nombre del tipo de evento, su clasificación y el Status Change o Time con el cual está relacionado. Para facilitar la definición de los Event Types, se considera importante incluir algunos ya precargados. En cualquier caso será una lista de tipos de eventos totalmente modificables por el usuario, que podrá añadir, editar o eliminar Event Types según su propio criterio. También se debe habilitar un botón que le permita volver al formulario anterior y, así mismo, otro para acceder al siguiente formulario.

El tercer y último formulario estará dedicado a la definición de los Event Characterization. Permitirá que se asocien uno o dos atributos a cada Event Type, y el resultado se mostrará en una lista. No se podrán borrar elementos de esta lista, puesto que todos los Event Types que están ligados a un Status Change deben caracterizarse. No así los que se relacionan con un Time concreto, que ni siquiera estarán listados en este formulario. Un botón permitirá, una vez se seleccione uno de los Event Characterization, editarlo. Los habituales botones que permiten ir adelante y atrás completan el formulario.

El orden de los formularios es inalterable, y de hecho no se han contemplado otras alternativas de diseño a ese respecto. El primer paso debe ser definir los Status, ya que los requieren las otras tres entidades que integran el módulo (Status Changes, Event Types y Event Characterizations). Antes de mostrar los Event Types es necesario saber qué Status Changes existen para poder mostrar su vinculación con éstos y guardarlos en la base de datos con sus claves ajenas correspondientes. Es por ello que comparten el primer formulario con los Status. Por último, los Event Characterizations aparecen en el último formulario del módulo debido a su dependencia con respecto a los Event Types, que deben configurarse previamente, en el formulario intermedio.

La sucesión de los formularios y los procesos que los unen pueden representarse gráficamente por medio de un diagrama de secuencia de UML, que aparece en la figura 5-4.

5.4 IMPLEMENTACIÓN DE LA APLICACIÓN

En esta sección se describe la implementación de la aplicación cuyo diseño se ha desarrollado anteriormente. Se trata de mostrar la apariencia de los formularios de la aplicación ya implementada, así como las funcionalidades asociadas a los mismos.

5.4.1 FORMULARIO DE DEFINICIÓN DE STATUS Y STATUS CHANGES

Tiene como función asistir al usuario en la definición de los Status y Status Changes que desea incluir en la nueva instancia. Los Status y Status Changes que se configuran en este formulario son válidos para todos los Infrastructure Access del Process Instance.

La lista de Status, que se sitúa a la izquierda, muestra tanto aquellos precargados en el software como los que ha definido el usuario. Junto a cada Status aparece su clasificación como Access Status o Allocation Status. Tres botones, justo a la derecha de esta primera lista, facilitan la posibilidad de añadir nuevos Status y modificar los existentes (lo cual se consigue por medio de un cuadro de diálogo programado para tales fines) o bien eliminarlos. En la figura 5-5 se muestra la apariencia de este primer formulario del módulo de Status. La figura 5-6 es una captura de pantalla del software en funcionamiento mientras se edita un Status.

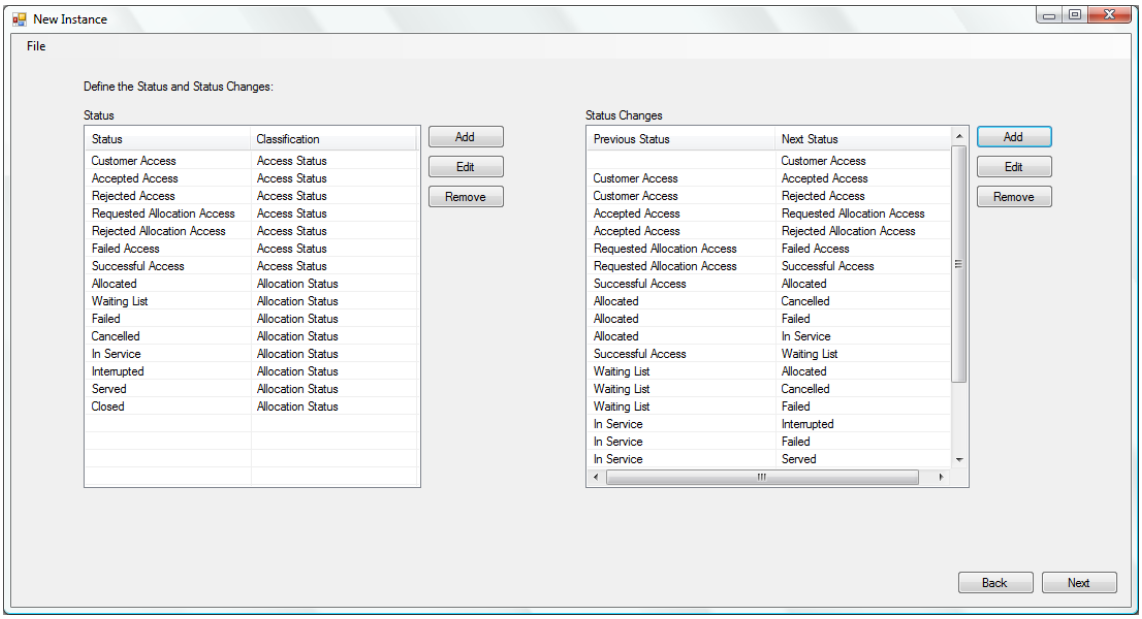


Figura 5-5. Primer formulario del módulo de definición de Status

Con el fin de facilitar la configuración de una instancia de proceso por parte de los usuarios y teniendo en cuenta qué Status serán útiles para operar algunos ejemplos en un hipotético simulador del nivel de ejecución, se han precargado a modo de plantilla los atributos que se detallan a continuación, en la tabla 5-3. No obstante estos Status son totalmente modificables por el usuario, por lo que el programa gana en usabilidad sin perder un ápice de flexibilidad.

Status	Clasificación
Customer Access	Access Status
Accepted Access	Access Status
Rejected Access	Access Status
Requested Allocation Access	Access Status
Rejected Allocation Access	Access Status
Failed Access	Access Status
Successful Access	Access Status
Allocated	Allocation Status
Waiting List	Allocation Status
Failed	Allocation Status
Cancelled	Allocation Status
In Service	Allocation Status
Interrupted	Allocation Status
Served	Allocation Status
Closed	Allocation Status

Tabla 5-3. Status preconfigurados en el primer formulario del módulo de definición de Status

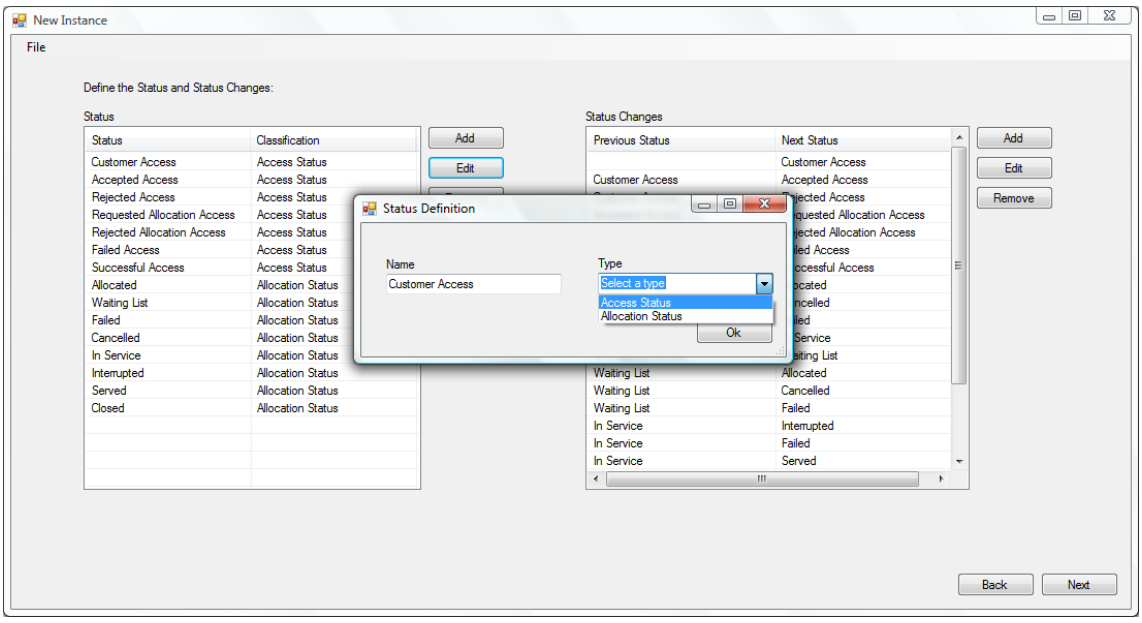


Figura 5-6. Cuadro de diálogo de adición y edición de Status

La lista de la derecha contiene los Status Changes que se aplican a todos los Infrastructure Access. Se actualiza automáticamente si se elimina alguno de los Status de la otra lista, de tal forma que se borran los Status Changes en los que dicho Status interviene, sea con el rol de Previous o con el de Next. Tres botones que se encuentran a

la izquierda de la lista permiten añadir un nuevo Status Change, editar uno de los existentes o bien eliminarlo. La adición y edición de elementos de la lista se realiza con la asistencia de un cuadro de diálogo que muestra dos combo boxes con todos los Status disponibles y permite al usuario elegir un ítem de cada una de ellas. Los Status Changes que se han precargado para facilitar la utilización del programa al usuario son los siguientes (tabla 5-4):

Previous Status	Next Status
	Customer Access
Customer Access	Accepted Access
Customer Access	Rejected Access
Accepted Access	Requested Allocation Access
Accepted Access	Rejected Allocation Access
Requested Allocation Access	Failed Access
Requested Allocation Access	Successful Access
Successful Access	Allocated
Allocated	Cancelled
Allocated	Failed
Allocated	In Service
Successful Access	Waiting List
Waiting List	Allocated
Waiting List	Cancelled
Waiting List	Failed
In Service	Interrupted
In Service	Failed
In Service	Served
Interrupted	In Service
Interrupted	Failed
Served	Closed
Cancelled	Closed
Failed	Closed

Tabla 5-4. Status Changes preconfigurados en el primer formulario del módulo de definición de Status

5.4.2 FORMULARIO DE DEFINICIÓN DE EVENT TYPES

El elemento central del formulario (figura 5-7) es una lista en la que el software muestra todos los Event Types que forman parte de la instancia del proceso. Junto al nombre de cada tipo de evento aparecen otras características, como su clasificación, el Status Change con el cual se relaciona o bien el Time al cual se halla vinculado.

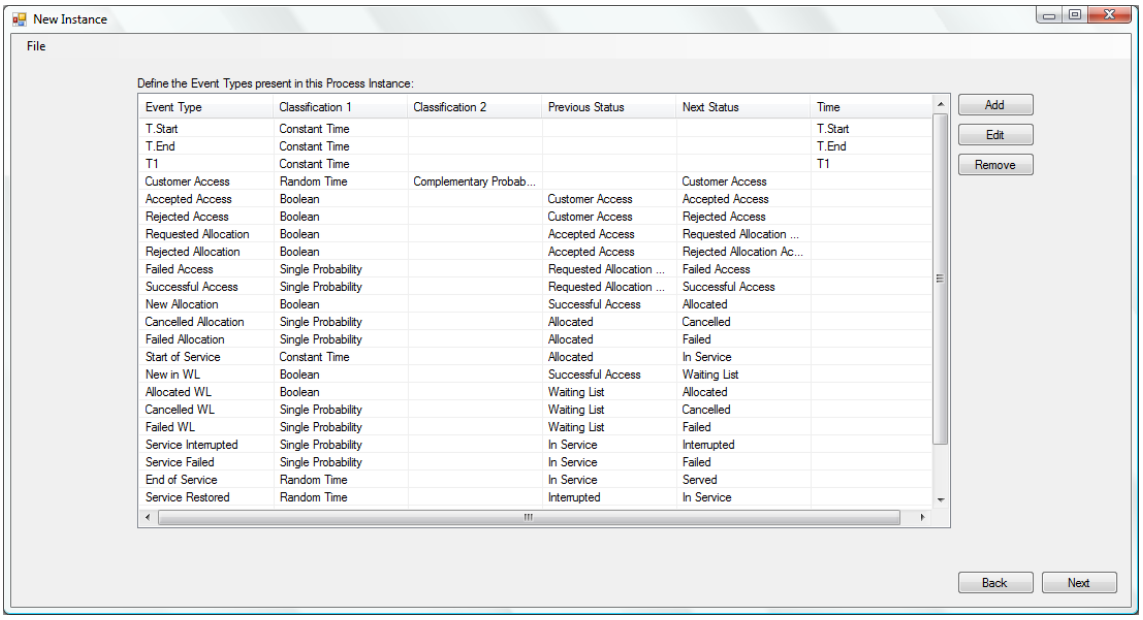


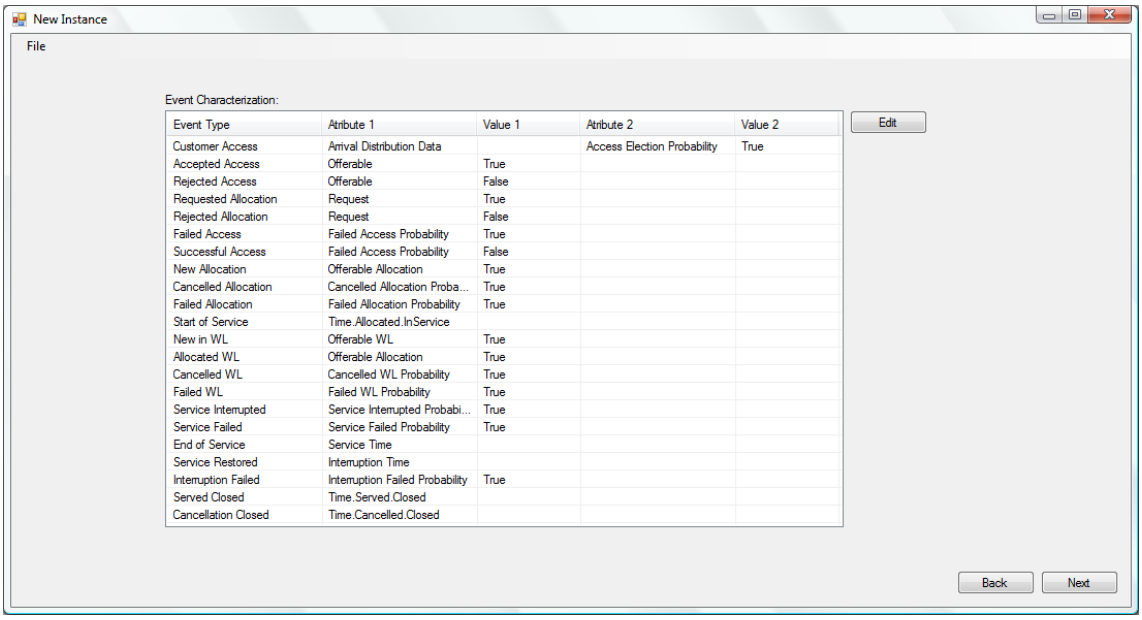
Figura 5-7. Formulario de adición y edición de Event Types del módulo del Process Instance Dynamics

La lista se actualiza automáticamente, eliminando de la lista un Event Type si el usuario borra un Status o un Status Change relacionados con él, o sincronizando los nombres de los Status si se producen cambios en el formulario anterior. Los eventos precargados son uno por cada Time y otro por cada Status Change. El nombre por defecto que se le da a los primeros es el del Time correspondiente, y a los segundos el del Status con rol de Next. Cualquier Event Type puede ser modificado por el usuario en cualquier momento con la ayuda de un cuadro de diálogo que surge cuando se pincha sobre el botón Edit. Este mismo cuadro de diálogo es el que permite al usuario incluir nuevos Event Types en su Process Instance tras pulsar el botón Add. Los Event Types se eliminan de la lista a través del botón Remove, una vez se ha seleccionado el que se quiere borrar.

5.4.3 FORMULARIO DE DEFINICIÓN DE EVENT CHARACTERIZATION

El último formulario del módulo de definición de la dinámica de Process Instance (figura 5-8) asiste al usuario en la caracterización de los Event Type relacionándolos con los atributos correspondientes y, en algunos casos, con el valor que deberán adoptar sus parámetros de atributo para que se dé cada tipo de evento.

La estructura es muy parecida a la del formulario anterior. Sin embargo hay una diferencia en cuanto a los Event Types que se listan, ya no se muestran aquellos ligados a un Time, sino sólo los vinculados a un Status Change.



Event Type	Atribute 1	Value 1	Atribute 2	Value 2
Customer Access	Arrival Distribution Data		Access Election Probability	True
Accepted Access	Offerable	True		
Rejected Access	Offerable	False		
Requested Allocation	Request	True		
Rejected Allocation	Request	False		
Failed Access	Failed Access Probability	True		
Successful Access	Failed Access Probability	False		
New Allocation	Offerable Allocation	True		
Cancelled Allocation	Cancelled Allocation Proba...	True		
Failed Allocation	Failed Allocation Probability	True		
Start of Service	Time.Allocated.InService			
New in WL	Offerable WL	True		
Allocated WL	Offerable Allocation	True		
Cancelled WL	Cancelled WL Probability	True		
Failed WL	Failed WL Probability	True		
Service Interrupted	Service Interrupted Probabi...	True		
Service Failed	Service Failed Probability	True		
End of Service	Service Time			
Service Restored	Interruption Time			
Interruption Failed	Interruption Failed Probability	True		
Served Closed	Time.Served.Closed			
Cancellation Closed	Time.Cancelled.Closed			

Figura 5-8. Formulario de caracterización de Event Types del módulo del Process Instance Dynamics

En las figuras 5-9 y 5-10 se muestra los diagramas de estados en UML del modelo de dinámica de Process Instance que el software tiene configurado por defecto, si bien es completamente modificable por el usuario en tiempo de ejecución.

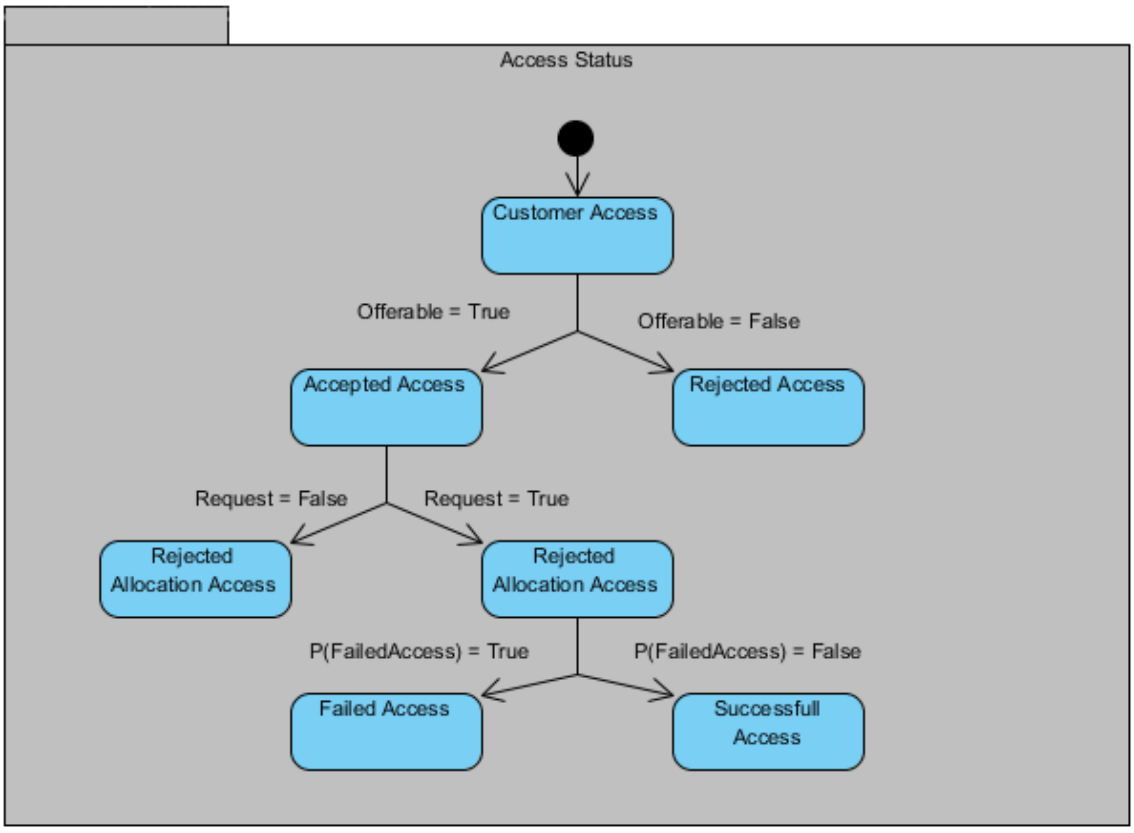


Figura 5-9. Ejemplo de diagrama de estados en UML de Access

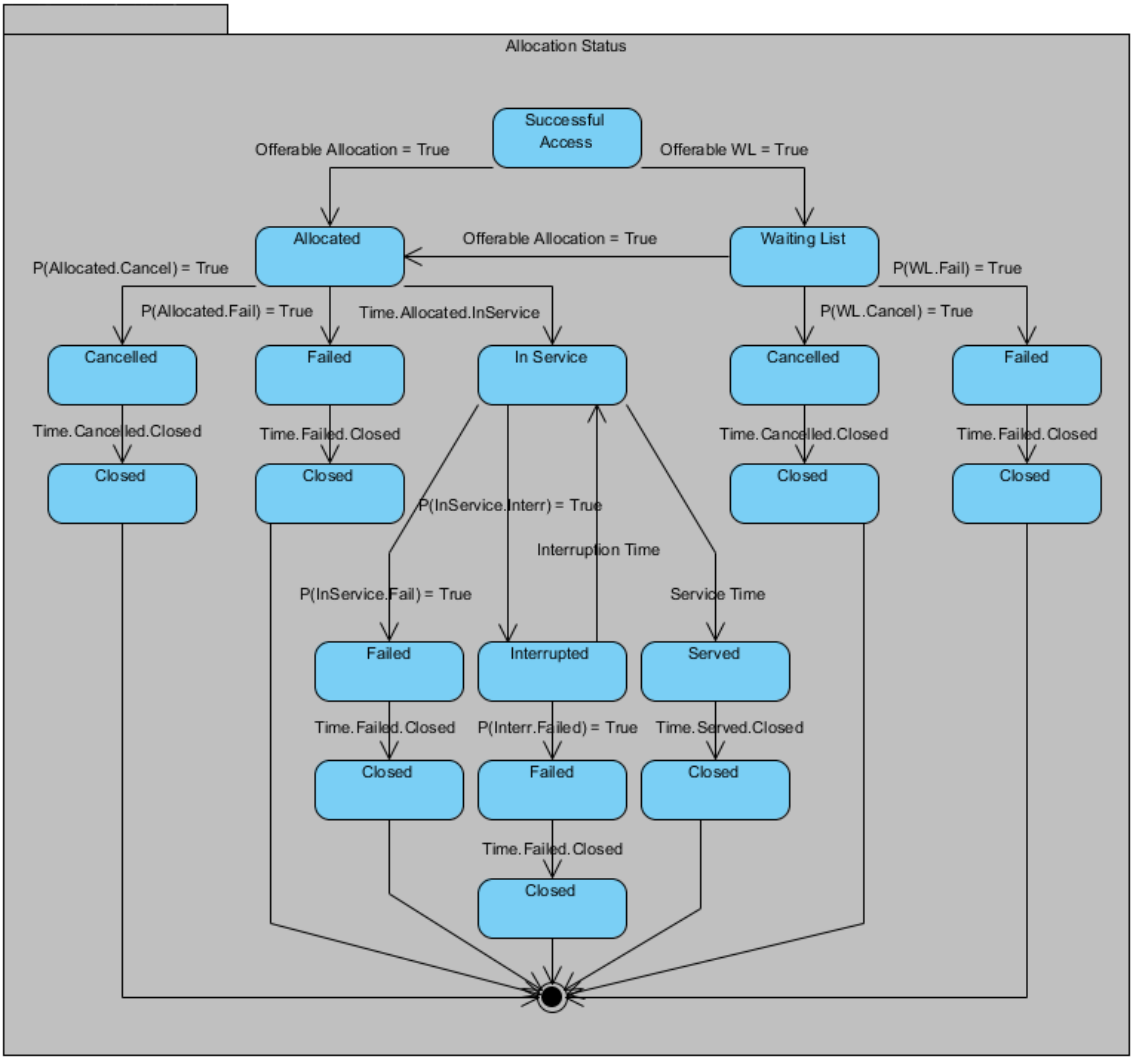


Figura 5-10. Ejemplo de diagrama de estados en UML de Allocation

6

MÓDULO DE DEFINICIÓN DE LA FUNCIÓN OBJETIVO

Tal y como se anunció en la Introducción y siguiendo la misma estructura que en los tres capítulos previos, en primer lugar se expone el diseño conceptual del modelo de definición de la función objetivo. Además, se analiza el diseño del segmento de la base de datos relacionada con la función objetivo, con la ayuda de un diagrama del modelo de datos. El diseño de la aplicación ocupa el tercer apartado del capítulo, mientras que el cuarto versa sobre la implementación del módulo del software correspondiente a la configuración de una función objetivo en el marco del planteamiento de un problema de gestión del ingreso (Revenue Management) con asignación de infraestructuras.

6.1 DISEÑO CONCEPTUAL

Ya se ha adelantado en la introducción de este capítulo que el primer apartado corresponde al diseño conceptual del modelo de definición de la función objetivo de un problema de gestión de ingreso (Revenue Management) con gestión de infraestructuras.

Una función objetivo (Objective Function) es una expresión matemática, en general multivariante, que se pretende optimizar, es decir, minimizarse o maximizarse. Hasta ahora, en los módulos anteriores nos hemos ocupado de posibilitar el establecimiento de las restricciones del problema. Sin embargo, la función objetivo no sólo forma parte de la definición del problema, sino que es un elemento esencial. Por un lado, dos problemas prácticamente idénticos, aunque sólo se diferencien en la función objetivo, tendrán soluciones, en general, distintas. Así mismo, la forma de la función objetivo puede condicionar la utilización de un método u otro de resolución. Si bien en este proyecto no se aborda la resolución de los problemas que se plantean, debido a una cuestión de alcance, sí se tiene en cuenta la posible inclusión del mismo en una línea de trabajo que se encargue de esa cuestión en trabajos posteriores.

En la figura 6-1 puede apreciarse un diagrama de clases en UML del modelo de la función objetivo. El diagrama de la figura 6-2 completa el modelo con el desarrollo de los componentes dinámicos de la función objetivo (Dynamics Components).

Así pues, cada instancia de un proceso de negocio (Process Instance) cuenta siempre con una, y sólo una, función objetivo (Objective Function). Es decir, se trata de una relación uno a uno. Una función objetivo, como hemos mencionado anteriormente, se puede querer minimizar o maximizar. Esta información se almacena en el atributo denominado Optimization (optimización). Para ello, contamos con una enumeración de los posibles valores que puede adoptar el atributo: minimización (Min) y maximización (Max).

Una función objetivo se compone de términos. Por tanto, se define un término como un elemento de la función objetivo, desde el núcleo más básico hasta la propia función objetivo. Por ejemplo, disponemos de la siguiente función objetivo:

$$F = x + 3 \cdot z$$

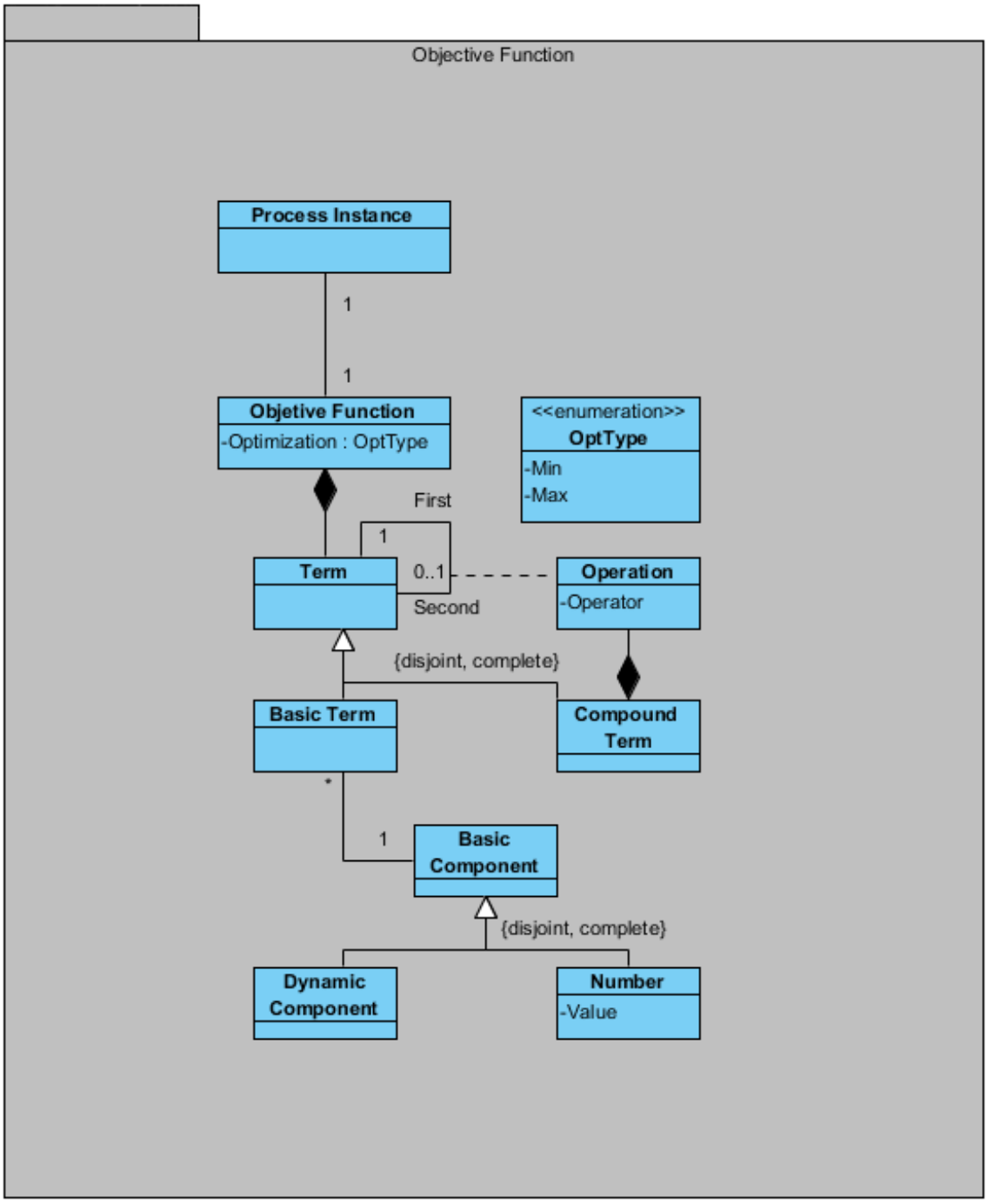


Figura 6-1. Diagrama UML del modelo de función objetivo

Si analizamos su composición, percibimos que el primer término es la propia función objetivo, es decir, $x + 3 \cdot z$. A continuación, separamos dicho término en otros dos términos, por un orden ascendiente de preferencia de operación. Es decir, si la suma es la última operación que se debe realizar para obtener un valor de la función objetivo cuando las variables adquieren un valor concreto, sepamos la función en dos términos separados por el símbolo de adición. Así pues, el segundo y tercer términos son,

respectivamente, *x* y *3 · z*. El segundo término no puede descomponerse más, pues no contiene ningún símbolo de operación. En cambio, el tercer término sí puede separarse en un cuarto y quinto términos separados por el símbolo de multiplicación: *3* y *z*.

ID del Término	Término
1	*x + 3 · z*
2	*x*
3	*3 · z*
4	*3*
5	*Z*

Tabla 6-1. Ejemplo: términos de una función objetivo

Los términos pueden ser básicos (Basic Terms) o compuestos (Compound Term). Esta clasificación es completa y disjunta. Un término es básico si no puede descomponerse en otros términos más simples. Utilizando el ejemplo anterior y la información contenida en la tabla 6-1, los términos con los identificadores 2, 4 y 5 son básicos. Con los términos compuestos ocurre exactamente lo contrario. Se componen de operaciones. Así pues, todo término compuesto está asociado directamente a una operación (Operation). Los términos con los identificadores 1 y 3 son términos complejos.

Una operación es una expresión que deriva de la relación entre dos términos, básicos o no y con los roles de primer (First) y segundo (Second) miembros, con la mediación de un operador (Operator). A su vez, como ya hemos mencionado, una operación es la forma de descomponer un término compuesto. Siguiendo con el ejemplo anterior, en la tabla 6-2 se muestra las dos operaciones que forman parte de la función objetivo:

ID de Operación	Término Compuesto	Primer Miembro	Operador	Segundo Miembro
1	*x + 3 · z*	*x*	*+*	*3 · z*
2	*3 · z*	*3*	*·*	*z*

Tabla 6-2. Ejemplo: operaciones en una función objetivo

La descomposición de cada término compuesto en operaciones continúa hasta que todos los términos son básicos. Cada término básico hace referencia a un único componente básico (Basic Component), como por ejemplo son el número *3*, o la variable *x*. Sin embargo, un mismo componente básico pudiera aparecer en más de una

ocasión en la función objetivo, por lo que en ese caso tendríamos distintos términos básicos que hacen referencia a un mismo componente básico. De ahí que la relación entre Basic Component y Basic Term sea de uno a muchos.

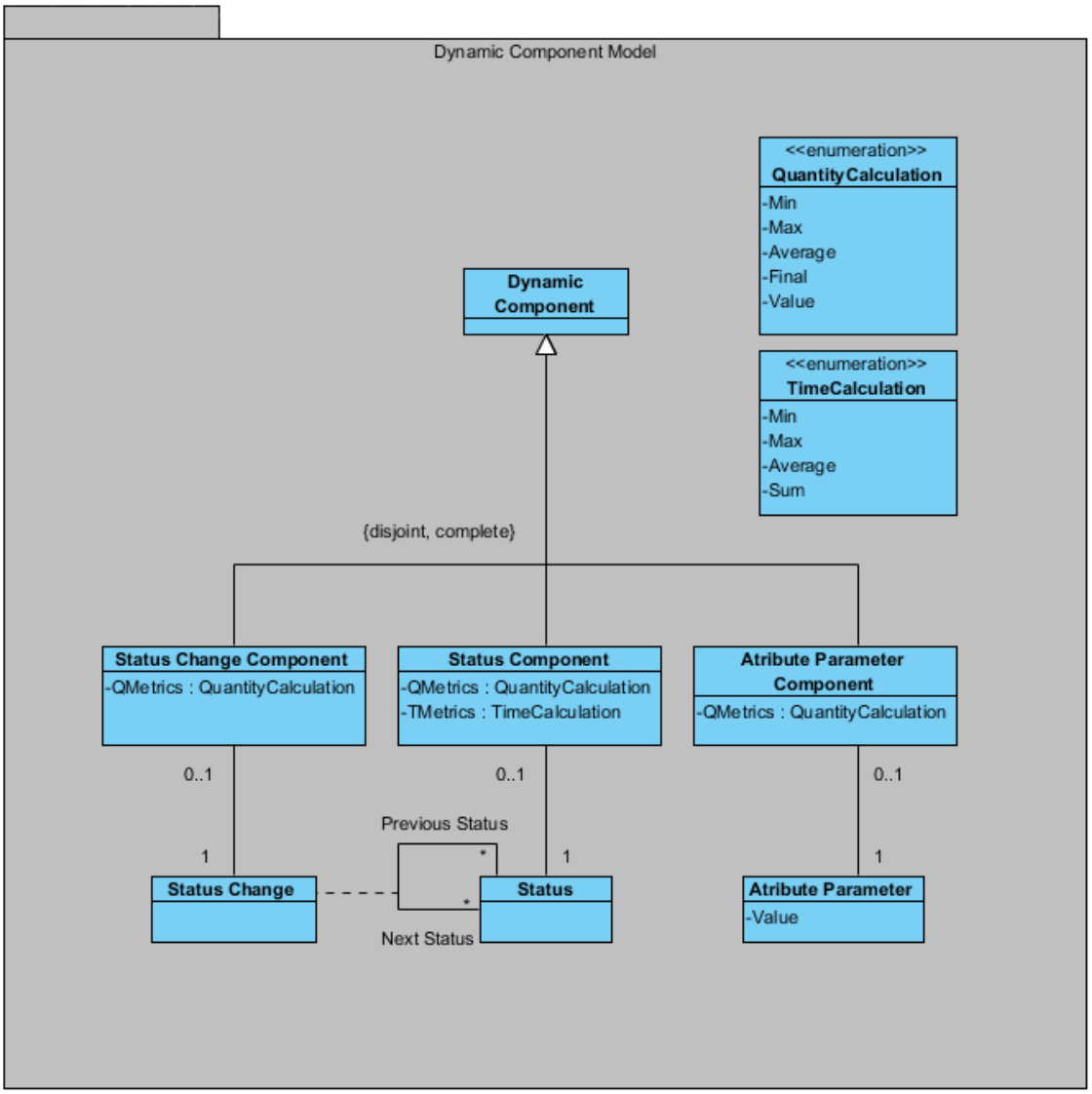


Figura 6-2. Diagrama UML del modelo de componente dinámico

Continuando con la explicación del modelo de función objetivo, los componentes básicos pueden clasificarse de forma completa y disjunta en componentes dinámicos (Dynamic Components) y números (Numbers). Los componentes dinámicos se explicarán pormenorizadamente a continuación. En lo que respecta a los números, el propio valor numérico se almacena en un atributo llamado Value. En nuestro ejemplo, *x* y *z* serían componentes dinámicos y *3*, evidentemente, un número.

En la figura 6-2 se muestra el modelo del componente dinámico de una función objetivo. Un componente dinámico es un componente básico cuyo valor puede depender bien del resultado de la ejecución de la instancia de proceso de negocio o bien de los valores otorgados a los parámetros de atributos en la fase de definición del problema. El rasgo diferencial más importante de los componentes dinámicos es que no son meras cifras, sino que tienen carga conceptual. Ejemplos de componentes dinámicos son el número de habitaciones dobles que tiene un hotel, el tiempo medio de espera de todos los pacientes en la unidad de cardiología de urgencias de un hospital o el número de reservas canceladas en una compañía de alquiler de coches por parte de clientes mayores de 25 años.

Los componentes dinámicos (Dynamic Components), pueden clasificarse de forma disjunta y completa en componentes de parámetro de atributo (Attribute Parameter Components), componentes de estado (Status Components) y componentes de cambio de estado (Status Change Components).

Los componentes de parámetro de atributo son componentes dinámicos cuyo valor está relacionado en cierta forma el de un determinado parámetro de atributo. Este parámetro de atributo puede pertenecer a una variable, a un parámetro dinámico o a un parámetro estático. En caso de que pertenezca a una variable o a un parámetro dinámico, se podrán elegir medidas de cantidad (Quantity Calculations), tales como el valor mínimo (Min), máximo (Max), medio (Average) o final (Final) que alcanza el parámetro de atributo en la ejecución de la instancia de proceso de negocio. Si el parámetro de atributo pertenece a un parámetro estático, tan sólo se podrá adoptar como medida el propio valor del parámetro de atributo (Value). El número de habitaciones dobles que tiene un hotel es un ejemplo de componente de parámetro de atributo que puede aparecer en una función objetivo.

Un componente de estado es un componente dinámico cuyo valor definitivo se asocia a un estado concreto, bien sea por medio de una medida de cantidad (Quantity Calculation), como el mínimo (Min), máximo (Max), número medio (Average) o número final (Final) de accesos o asignaciones de un tipo que se encuentran en un determinado estado; o bien a través de una medida de tiempo (Time Calculation), como el tiempo mínimo (Min), máximo (Max), medio (Average) o total (Sum) que un tipo concreto de accesos o asignaciones pasa en un determinado estado durante la ejecución de la instancia de proceso de negocio. Un ejemplo es el tiempo medio de espera de todos los pacientes en la unidad de cardiología de urgencias de un hospital.

Un cambio de estado puede relacionarse con un componente de cambio de estado. El valor que adopta este componente en la función objetivo depende del resultado de la ejecución de la instancia de proceso y de la medida de cantidad elegida (Quantity Calculation), el mínimo (Min), máximo (Max), número medio (Average) o número final (Final) de cambios de estado determinados de un tipo de acceso o asignación. Como ejemplo de componente de cambio de estado podemos señalar el número de reservas canceladas en una compañía de alquiler de coches por parte de clientes mayores de 25 años.

6.2 MODELO DE DATOS

El modelo de datos del módulo de definición de la función objetivo es el tema de esta sección. Para exponerlo, nos serviremos del diagrama IDEF1X elaborado con ERWin que aparece en la figura 6-3, y en el cual puede apreciarse el diseño lógico del segmento de la base de datos correspondiente al mencionado módulo.

La primera decisión clave es la de decidir qué tablas se necesitan. Se considera cada una de las entidades que aparecen en los diagramas de clases UML representados en la figuras 6-1 y 6-2. Las entidades que requieren una tabla propia son la función objetivo (Objective Function), el término (Term), el término básico (Basic Term), el término compuesto (Compound Term), la operación (Operation) , y el componente básico (Basic Component), tal y como puede apreciarse en la figura 6-3.

Cuando se comienza a definir una función objetivo en el marco de una instancia de proceso de negocio, se empieza por inscribir a la misma en la tabla ObjectiveFunction. La clave primaria de esta tabla es en parte heredada de la tabla ProcessInstance (ID_ProcessModel e ID_ProcessInstance) y en parte propia (ID_ObjectiveFunction). Además la tabla cuenta con un atributo denominado Optimization, que permite al usuario guardar el tipo de optimización que se debe aplicar a la función (maximización o minimización).

Los términos de la función objetivo se almacenan en la tabla Term. La clave de la tabla Term es parcialmente heredada. Recibe de ObjectiveFunction ID_ProcessModel, ID_ProcessInstance e ID_ObjectiveFunction. Además cuenta con una clave propia que forma parte de la clave principal, llamada ID_Term. El atributo Basic de la tabla Term permite clasificar los registros de la misma (es decir, los términos), como términos básicos o compuestos en función del valor de dicho atributo en cada registro.

Por otro lado, la tabla BasicComponent, que es la de diseño más complejo de este módulo, recibe claves ajenas de tres tablas: AttributeParameter, TableStatus y StatusChange. De todas ellas hereda, como parte de la clave principal, ID_ProcessModel e ID_ProcessInstance. La clave principal se completa con una clave propia, ID_BasicComponent. Ya como claves ajenas que no forman parte de la clave principal, BasicComponent recibe ID_InstanceEntity, ID_Attribute y AttributeParameter de AttributeParameter; ID_Status de TableStatus; y por último ID_PreviousStatus e ID_NextStatus de StatusChange. Estas seis claves ajenas permiten identificar de forma unívoca un registro si corresponde con un componente dinámico, sea del tipo que sea. Seis atributos completan el diseño de la tabla. El primero, Dynamic, indica si el componente es dinámico o bien se trata de un número. Si se trata de un componente dinámico, su nombre se almacena en el atributo Name_, su tipo (de parámetro de atributo, de estado o de cambio de estado) se guarda en Type_ y la medida utilizada en los atributos QMetrics o TMetrics, según corresponda. En cambio, si es un número, simplemente su valor se recopila en el atributo Value_ y el resto de atributos no se rellena.

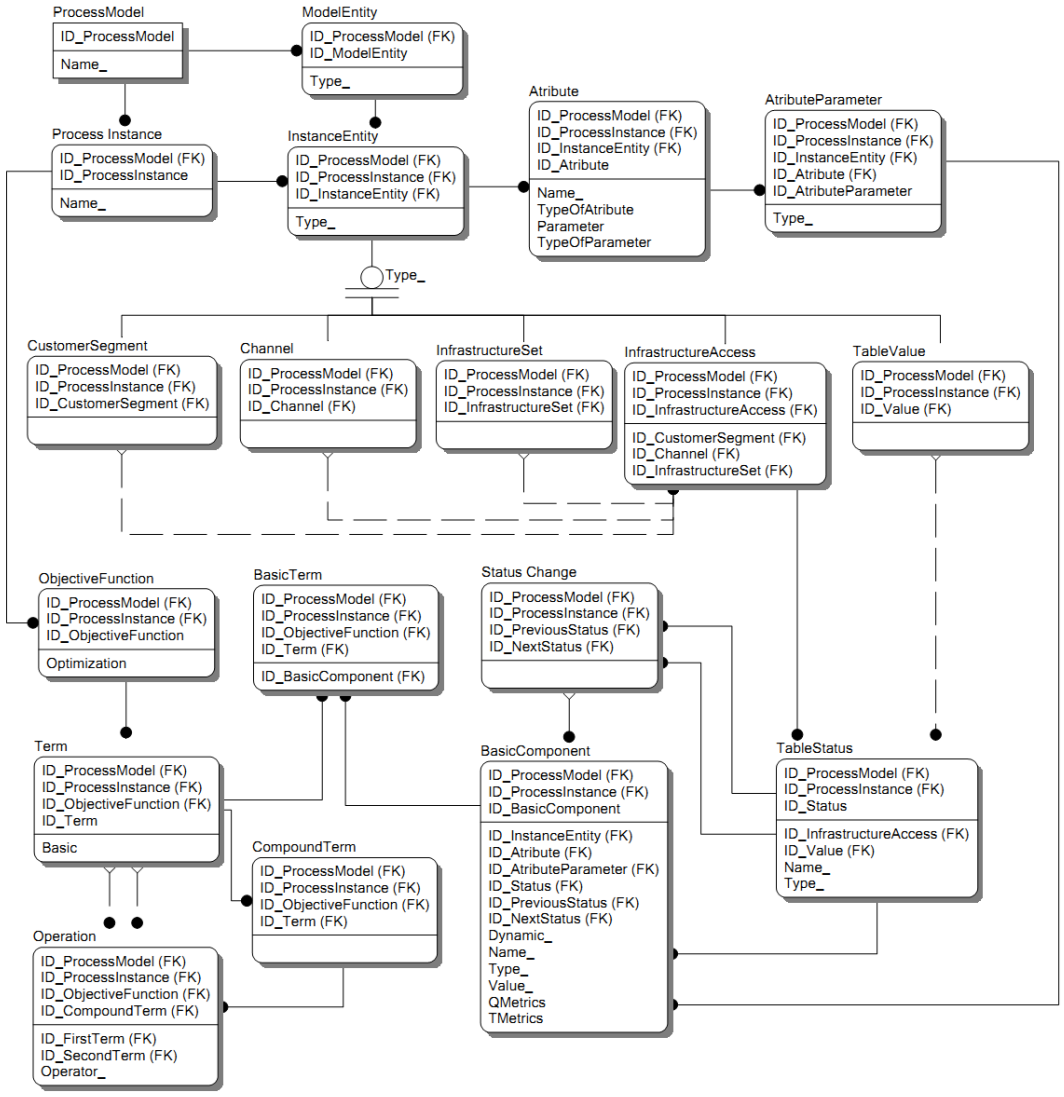


Figura 6-3. Diagrama IDEF1X del segmento de la base de datos correspondiente al modelo de función objetivo

El siguiente paso lógico del análisis es estudiar la tabla BasicTerm, ya que se relaciona únicamente con las dos últimas tablas explicadas. Su clave principal está formada únicamente por claves ajenas: ID_ProcessModel, ID_ProcessInstance, ID_ObjectiveFunction e ID_Term que enteramente de Term. Además, cada registro de la tabla BasicTerm, es decir, cada término básico, se relaciona necesariamente con un componente básico, por lo que cuenta con la clave ajena ID_BasicComponent, que no forma parte de la clave principal pero permite la asociación.

La tabla CompoundTerm se basa exactamente en el mismo diseño que la tabla BasicTerm, excepto por el hecho de que no se relaciona con la tabla BasicComponent y por tanto no cuenta con la clave ajena ID_BasicComponent. Incluir las tablas

BasicTerm y CompoundTerm en el diseño no fue una decisión inmediata, puesto que el prescindir de ambas y contar tan sólo con la tabla Term era una alternativa factible. El motivo por el que finalmente se incluyó a ambas tablas fue el de explicitar las diferentes relaciones de los términos básicos y los términos compuestos con los registros de las tablas BasicComponent y Operation respectivamente.

La última tabla a explicar es Operation. Recibe de CompoundTerm todas las claves que forman su clave principal, es decir, ID_ProcessModel, ID_ProcessInstance, ID_ObjectiveFunction e ID_Term (esta última con la denominación de ID_CompoundTerm). Además, cada registro, que como hemos dicho va asociado a uno y sólo uno de los registros de la tabla CompoundTerm, cuenta con claves adicionales que no son parte de la clave principal: las dos son heredadas de sendos registros de Term con la denominación de ID_FirstTerm e ID_SecondTerm, y se corresponden con las claves de los cada uno de los dos términos que se relacionan por medio de un operador (almacenado en el atributo Operator) para dar lugar a una operación, es decir, a un registro de la propia tabla Operation.

6.3 DISEÑO DE LA APLICACIÓN

El proceso de diseño del módulo de la aplicación que asiste la definición de la función objetivo de una instancia de proceso es el tema de esta sección. Se desarrolla cuál es la estructura de formularios que sigue el módulo y se representa un diagrama de secuencia de UML (figura 6-4).

Se pretende que exista un formulario que centralice el proceso de definición de la función objetivo. Dicho formulario debe disponer de un display para mostrar en todo momento el estado de la función objetivo. La adición de términos, operadores y paréntesis puede realizarse mediante botones específicos, a modo de calculadora. Este formulario debe permitir también visualizar en todo momento cuáles son las propiedades de los componentes dinámicos que aparecen como términos de la función objetivo.

La adición de componentes básicos se puede articular por medio de cuadros de diálogo específico para cada tipo. Una de las opciones que es necesario incluir en la aplicación, y de la que no se ha hablado en el diseño de datos, es la posibilidad de que el usuario defina sumatorios de componentes básicos, algo imprescindible para que la configuración de la función objetivo sea un proceso cómodo para el usuario.

Si tenemos en cuenta que es posible que durante el proceso de definición de la función el usuario cometa errores, se piensa incluir dos mecanismos que permitan subsanarlos, ya sea comenzando de nuevo la definición o deshaciendo el último cambio.

Los formularios AddedAPComponent, AddedSComponent y AddedSCComponent muestran, cuando el usuario lo solicita, los componentes básicos de cada tipo ya añadidos.

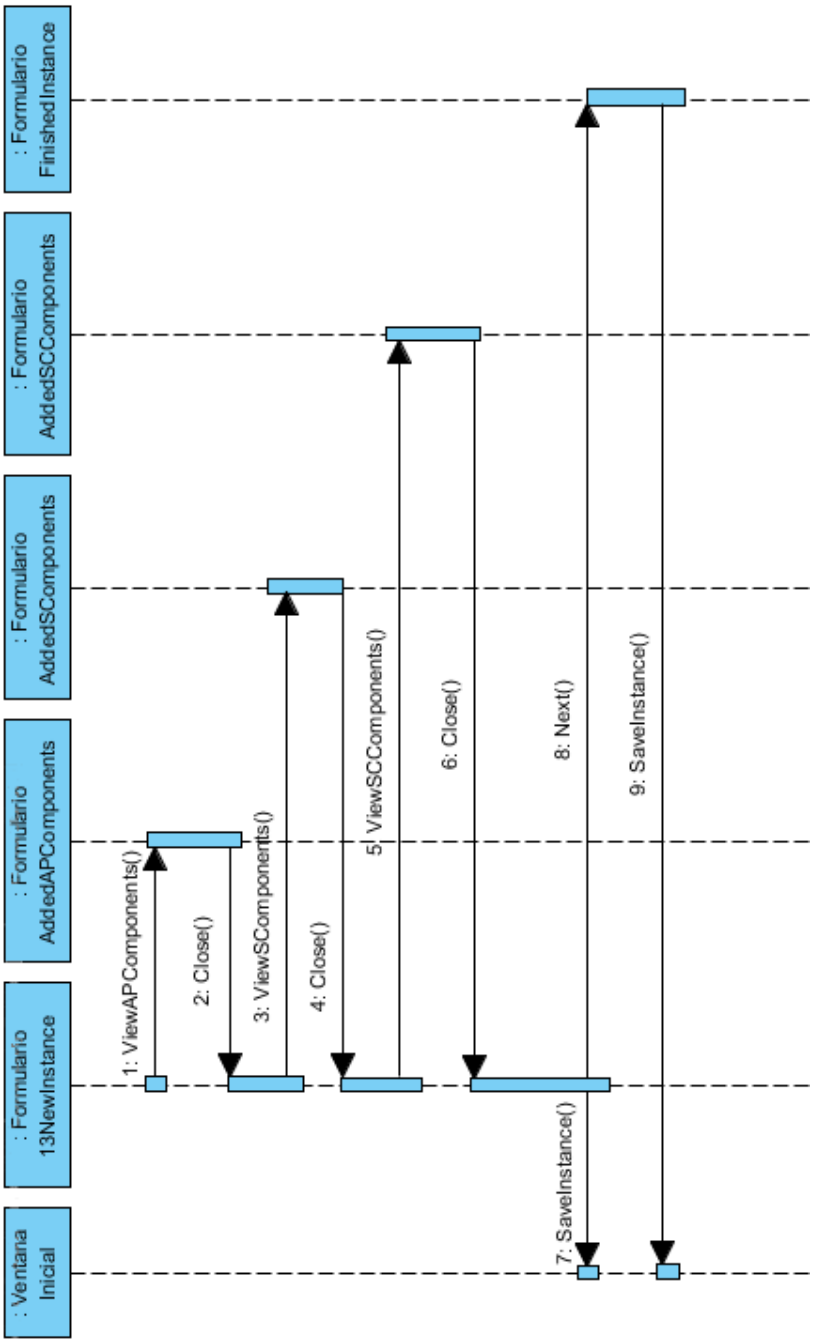


Figura 6-4. Diagrama de secuencia en UML del módulo de la aplicación asociado a la definición de la función objetivo

6.4 IMPLEMENTACIÓN DE LA APLICACIÓN

Esta sección versa acerca de la implementación de la aplicación cuyo diseño se ha expuesto en la sección anterior. Se pretende exponer la apariencia de los formularios de la aplicación ya implementada, así como las funcionalidades asociadas a los mismos.

6.4.1 FORMULARIO DE DEFINICIÓN DE LA FUNCIÓN OBJETIVO

Su función es la de asistir al usuario en la definición de la función objetivo. El layout del formulario puede apreciarse en la figura 6-5. La mayor parte del formulario lo ocupa un cuadro de texto cuya misión es la de almacenar y mostrar el estado actualizado de la función objetivo.

A la parte derecha del formulario se incluyen:

- Un combo box que permite elegir el tipo de optimización que se desea aplicar a la función objetivo
- Tres botones que dan la posibilidad de añadir componentes dinámicos a la función objetivo. Cada uno de los botones da paso a un cuadro de diálogo distinto que habilita al usuario para incluir un componente dinámico del tipo asociado al botón: sea un componente de parámetro de atributo (Add Attribute Parameter Component), de estado (Add Status Component) o de cambio de estado (Add Status Change Component).
- Un botón que muestra un cuadro de diálogo para incluir números en la función objetivo (Add Number)
- Cinco botones para añadir signos de operación a la expresión: suma (+), resta (-), multiplicación (*), división (/) y potencia (^).
- Dos botones para abrir y cerrar paréntesis
- Un botón de deshacer la última acción realizada (Undo)
- Un botón de borrado o reseteado (Clear all), para inicializar el formulario
- Tres botones (los tres que se hallan debajo del cuadro de texto y que comienzan con la palabra View) que permiten acceder cada uno a un formulario distinto en el que aparecen representados los componentes dinámicos del tipo asociado al botón.

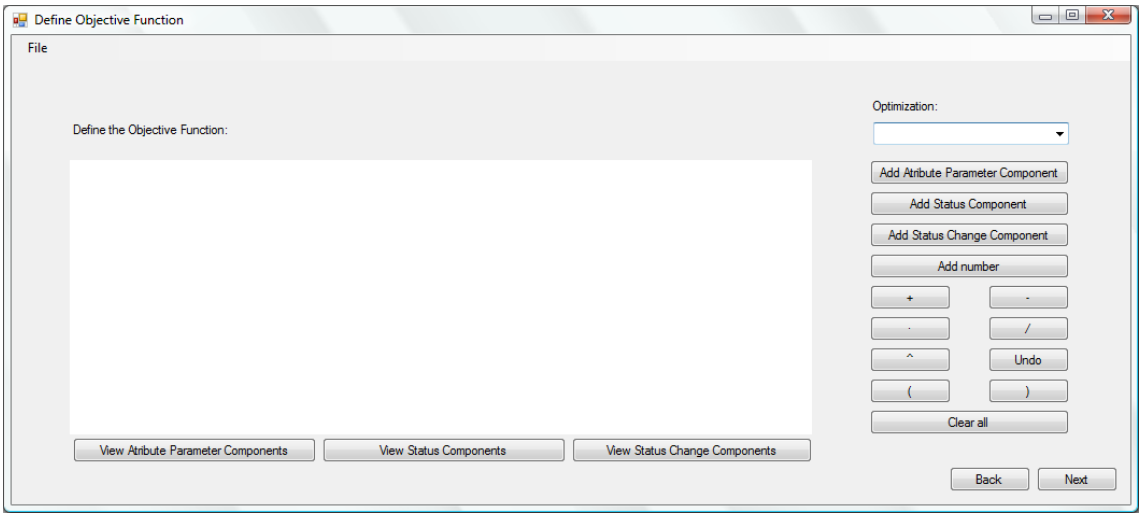


Figura 6-5. Formulario principal de definición de la función objetivo

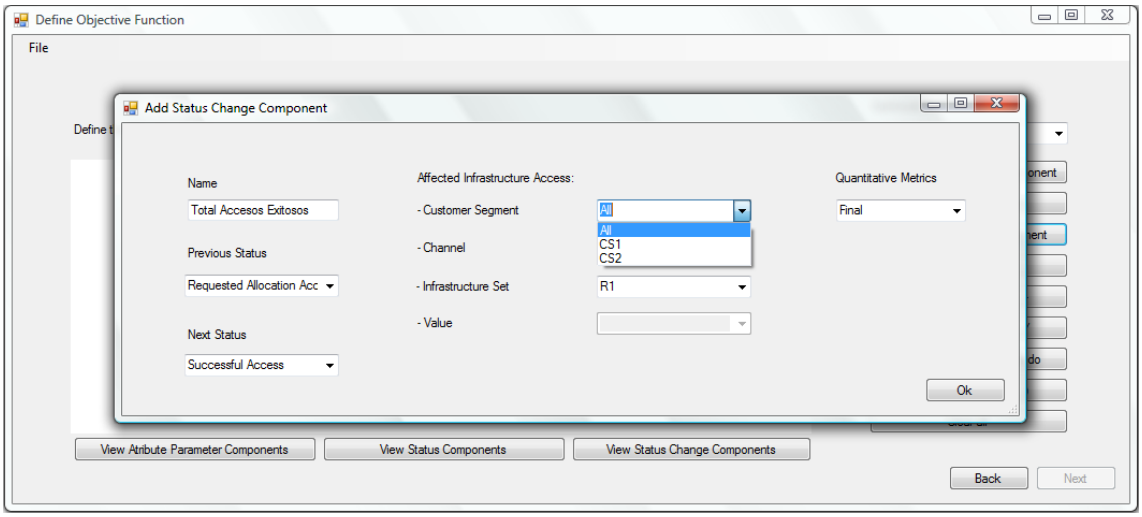


Figura 6-6. Cuadro de diálogo para la definición de un componente dinámico de cambio de estado

Por ejemplo, el cuadro de diálogo que aparece en la figura 6-6 es el que surge cuando el usuario desea incluir un componente dinámico de cambio de estado en su modelo. El programa le solicita en primer lugar un nombre, que será el que identificará al término asociado con el componente en el seno de la función objetivo, en la cual aparecerá entre corchetes y con los espacios sustituidos por barras bajas, con el fin de facilitar la identificación del mismo por parte del programa en los procesos de guardado y carga desde la base de datos.

A continuación, en el caso de los componentes dinámicos de cambio de estado, se deben completar los campos que solicitan el nombre del estado previo (Previous Status) y el estado próximo (Next Status). Si se tratase de un componente de parámetro de atributo, en esa parte del cuadro de diálogo se pediría el tipo de entidad a la que pertenece el atributo y de qué atributo se trata. Para los componentes de estado, se pide informar del tipo del estado (de acceso o de asignación) y de qué estado concreto se trata.

De cara a explicar qué información se solicita en la parte central y derecha del cuadro de diálogo, es necesario diferenciar por tipo de componente dinámico que se desea añadir:

- En el caso de los componentes de parámetros de atributo, según cuál sea el tipo de entidad al que pertenece el atributo, se activarán unos u otros combo boxes. Si se elige como tipo el segmento de clientes (Cliente Segment), se activará el combo box de segmento de clientes. Lo mismo ocurrirá, de manera análoga, para el canal (Channel), el conjunto de infraestructura (Infrastructure Set), el valor (Value) o el tiempo (Time). Si el usuario elige como tipo de entidad el acceso de una infraestructura, se activarán los combo boxes de segmento de clientes, canal y conjunto de infraestructura, ya que son los que permiten identificar unívocamente un acceso de infraestructura (Infrastructure Access). Si el tipo de entidad elegida es asignación

(Allocation), los combo boxes de segmento de clientes, canal, conjunto de infraestructura, valor y los dos de tiempo se mostrarán como activos.

Con el fin de introducir la posibilidad de que el usuario realice sumatorios de forma automática, basta con que en el combo box correspondiente se marque la opción "All". De esta forma, el software tendrá en cuenta para los cálculos a todos los elementos de esa categoría. Por ejemplo, en el caso de que se elija un atributo de Infrastructure Access y en el combo box de Customer Segment se seleccione la opción All, se sumará el valor del parámetro de atributo elegido de todos los Infrastructure Access que comparten Channel e Infrastructure Set, independientemente de su Customer Segment.

El valor que finalmente adquiere el componente en la función objetivo depende de la métrica de cantidad elegida. Si el parámetro de atributo pertenece a un atributo clasificado como parámetro estático, tan sólo puede adquirir el propio valor del parámetro (Value). Si estamos ante una variable o un parámetro estático, el software permitirá que se elijan cualquiera de los siguientes valores: mínimo, máximo, media y valor final. La elección se realiza con la asistencia del combo box de Quantitative Metrics.

- En el caso de los componentes de estado, si el estado seleccionado es de acceso (Access Status), se habilitarán los combo boxes del segmento de cliente, del canal y del conjunto de infraestructura. Por otra parte, si el estado elegido es de asignación (Allocation Status), los combo boxes del segmento de cliente, del canal, del conjunto de infraestructura y del valor aparecerán como activados. Los sumatorios funcionan de manera semejante a como lo hacen para los componentes de parámetro de atributo.

  En la expresión de la función objetivo, una vez finalizada la ejecución, el componente de estado adquiriría un valor concreto. Este valor puede ser una medida de cantidad (contador del número mínimo, máximo, medio o final de los accesos o asignaciones seleccionados en un estado determinado) o bien una medida de tiempo (tiempo mínimo, máximo, medio o suma de todos los tiempos de los accesos o asignaciones elegidos previamente en el estado seleccionado). Cuando el usuario elija una medida del combo box de Time Metrics, el combo box de Quantitative Metrics se desactivará, y viceversa.

- Finalmente, para la adición de componentes de cambio de estado, los combo boxes siguen las mismas reglas de activación, tomando como referencia el estado previo (Previous Status). También puede operarse con sumatorios, como en los dos casos anteriores.

  Los componentes de cambio de estado adquieren un valor en forma de media de cantidad, es decir como contador del número mínimo, máximo, medio o total de cambios de estado que se producen en la ejecución de la instancia de proceso. La elección de la medida se hace a través del combo box de Quantity Metrics.

6.4.2 FORMULARIOS DE COMPONENTES DINÁMICOS AÑADIDOS

El elemento clave de los tres formularios es una lista en la que el software muestra todos los componentes del tipo correspondiente que han sido definidos por el usuario para formar parte de la función objetivo, tal y como puede observarse en la figura 6-7.

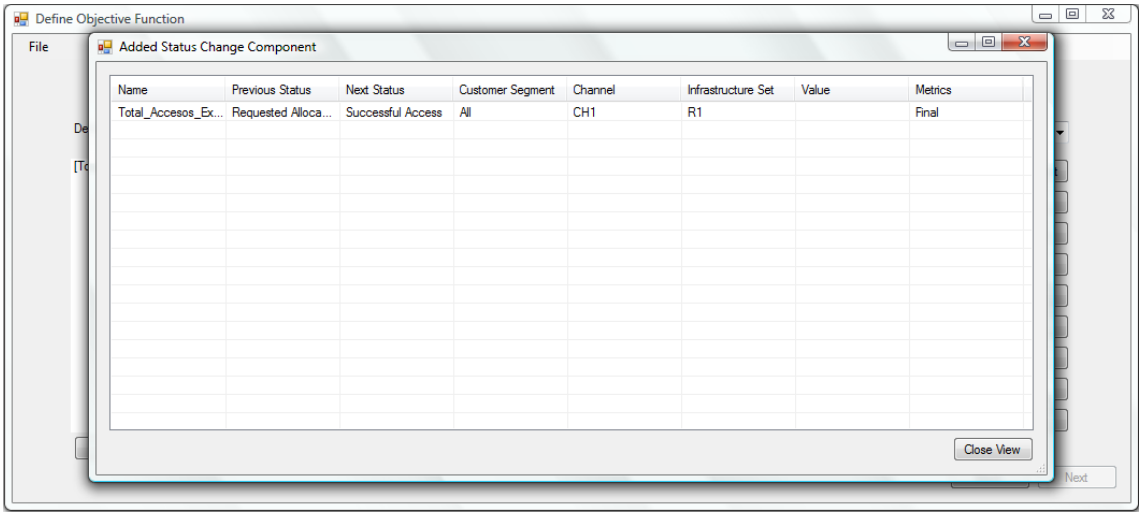


Figura 6-7. Formulario de consulta de los componentes dinámicos de cambio de estado añadidos

7

Ejemplo de Uso de la Herramienta

En este capítulo se presenta un ejemplo de aplicación de la herramienta informática en el planteamiento de un problema de asignación de infraestructuras en el sector sanitario, incluyendo definición del modelo y de una instancia de proceso.

7.1 EJEMPLO: UNIDAD DE CARDIOLOGÍA DE URGENCIAS

Un hospital considera que necesita reestructurar la Unidad de Cardiología de Urgencias debido a que el presupuesto del año próximo será más ajustado. Sin embargo, no se desea que este incremento de la eficiencia vaya en perjuicio de la calidad del servicio ofrecido a los pacientes. Por ello, el equipo de gestión del hospital se decide a utilizar nuestra herramienta informática para plantear su problema.

Uno de los analistas que trabaja para el equipo de gestión ejecuta el RM Modeler 1.0 y, puesto que es la primera vez que utiliza este software para modelar el proceso de la Unidad de Cardiología, necesita crear un nuevo modelo y darle nombre (figura 7-1).

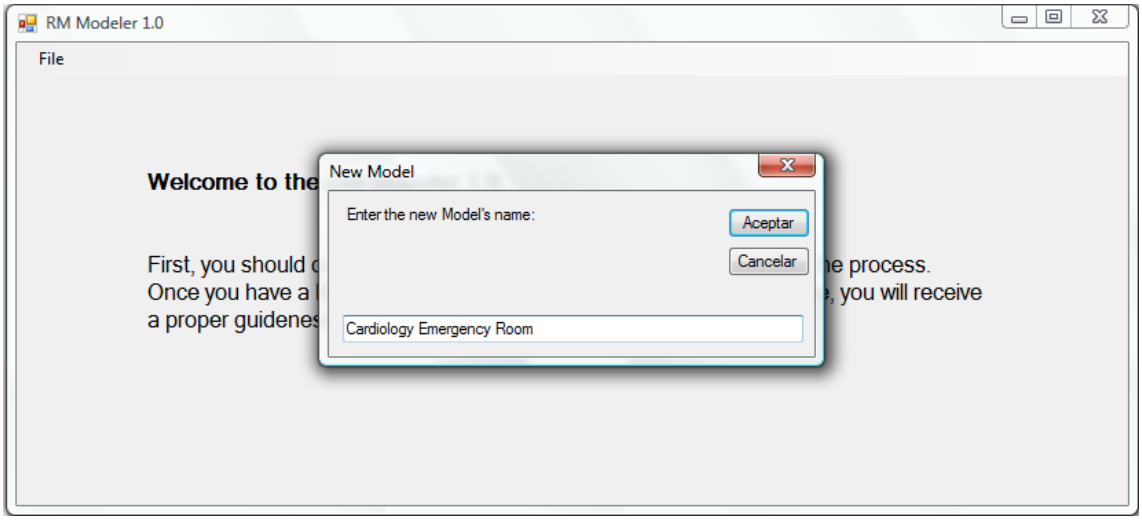


Figura 7-1. Ejemplo del Sector Sanitario: Creación de un Process Model

A continuación, la herramienta le permite establecer cuáles son los tipos de segmento de cliente, tipos de canal, tipos de infraestructura y tipos de valor que tienen cabida en el modelo (figura 7-2). Con respecto a los pacientes o clientes, se les segmenta por su gravedad. Así pues, se identifican dos tipos de segmento de cliente: levemente enfermos (Slightly ill) y gravemente enfermos (Seriously ill). Sobre los tipos de canal, los enfermos acceden a la Unidad de Cardiología de Urgencias bien en ambulancia (Ambulance) o bien por su propio pie, a través de la recepción del hospital (Reception). En la Unidad hay dos tipos de médicos, los cardiólogos (Cardiologists), médicos que han finalizado su residencia y han cursado la especialidad con éxito, y los médicos internos residentes (Medical Intern Residents), que se encuentran en proceso de

formación. Con respecto a los tipos de valor, se identifican dos, alto (High) y bajo (Low). Estos valores se utilizarán como medida del beneficio social (un concepto algo abstracto) que supone que se lleve a cabo con éxito la atención a un paciente de una forma o de otra.

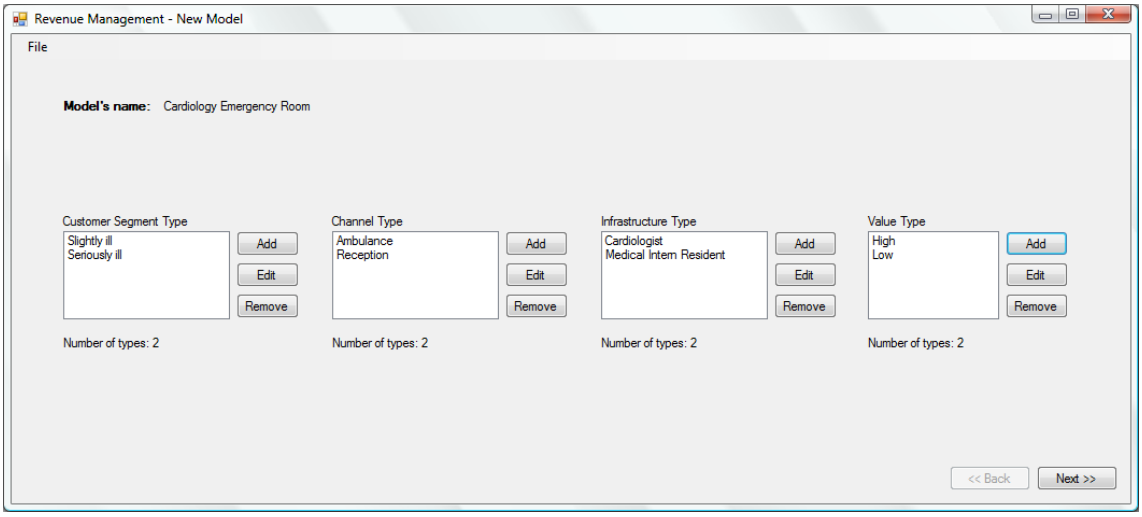


Figura 7-2. Ejemplo del Sector Sanitario: Definición de Customer Segment Types, Channel Types, Infrastructure Types y Value Types

El siguiente paso consiste en elegir cuáles de las posibles combinaciones de de tipos de segmentos de cliente, tipo de canal y tipo de infraestructura se constituyen como tipos de acceso a una infraestructura (Infrastructure Access Type). La configuración elegida por el equipo gestor se muestra en las figuras 7-3 y 7-4.

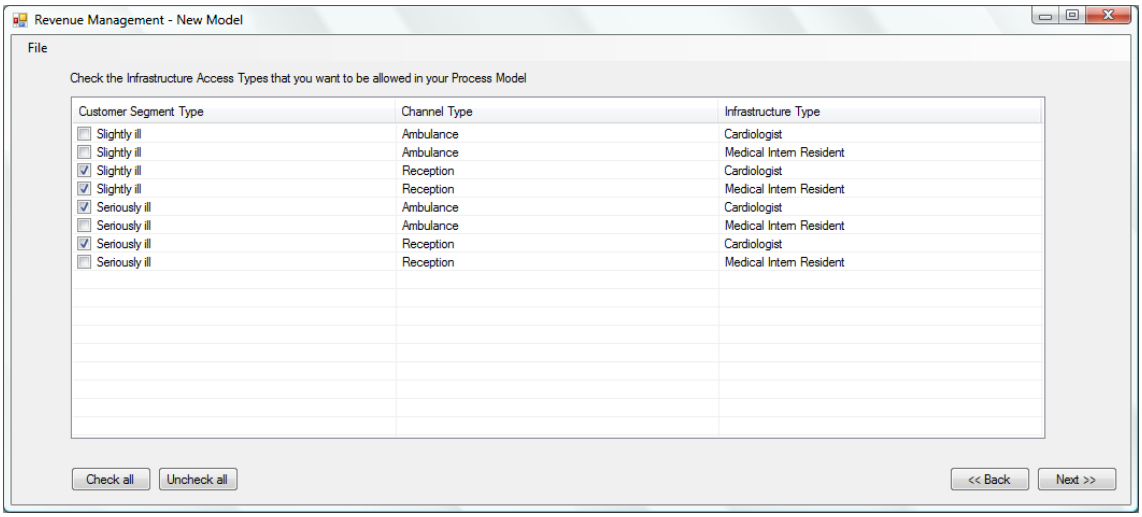


Figura 7-3. Ejemplo del Sector Sanitario: Proceso de configuración de los Infrastructure Access Types

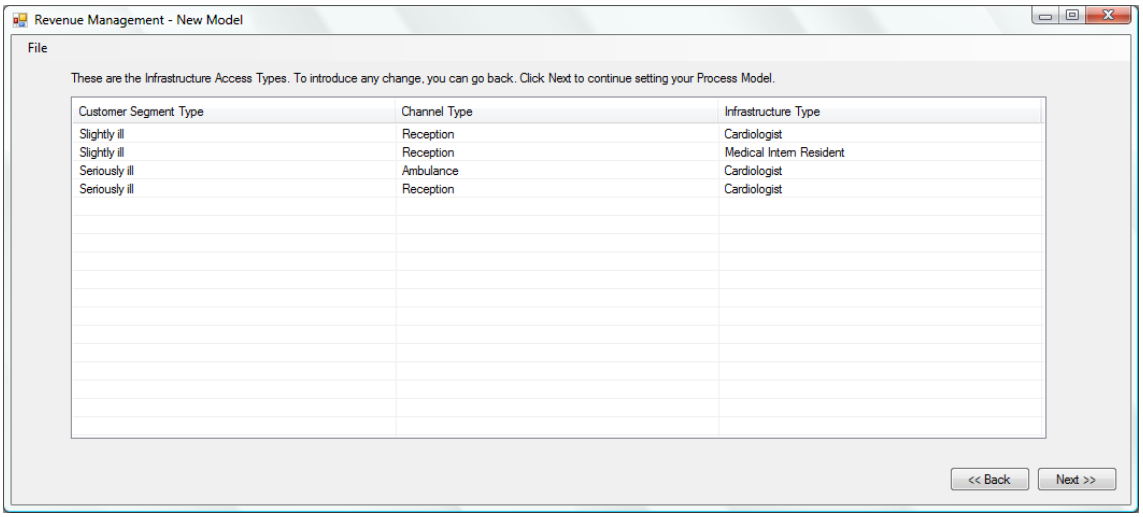


Figura 7-4. Ejemplo del Sector Sanitario: Configuración definitiva de los Infrastructure Access Types

Con respecto a los Infrastructure Access Types elegidos, caben algunas consideraciones. En primer lugar, se restringe la posibilidad de que un paciente que se encuentra en estado grave sea atendido por un médico interno residente, que está aún en formación. Se prefiere que este tipo de pacientes sean atendidos sólo por los cardiólogos, para asegurar una atención de la máxima calidad a los enfermos en una situación tan delicada. Por otra parte, no se considera la posibilidad de que un paciente con una dolencia leve llegue en ambulancia, puesto que la ambulancia sólo da cobertura a enfermos de una cierta gravedad.

En lo que respecta a los tiempos genéricos, se requiere un tiempo genérico de inicio y un tiempo genérico de fin, sin necesidad de tiempos genéricos intermedios. Las restricciones son triviales (figura 7-5).

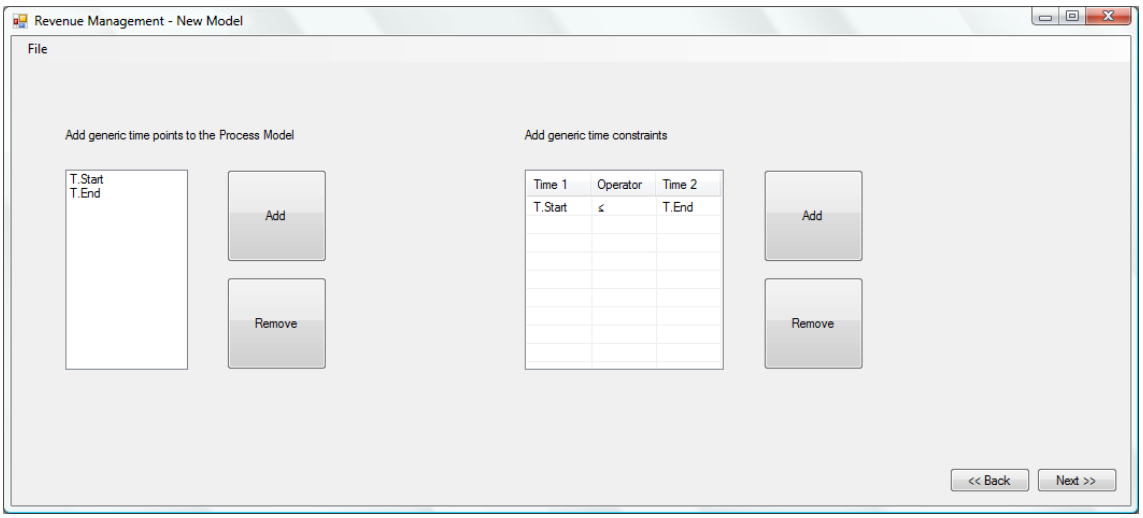


Figura 7-5. Ejemplo del Sector Sanitario: definición de los Generic Times y sus Constraints

El siguiente y último paso de la definición del modelo de proceso consiste en establecer los Allocations Types que forman parte del mismo (figura 7-6). Se consideran los cuatro Infrastructure Access Types existentes y se les asigna un tipo de valor y un comienzo y fin genéricos.

Se considera que todos los Allocation Types generan un beneficio social alto, excepto el que implica que los enfermos leves puedan ser atendidos por un cardiólogo. Se debe a que atender satisfactoriamente a un paciente grave siempre es valioso, mientras que en el caso de un paciente leve sólo se trata de un éxito comparable si supone una oportunidad para la formación del médico interno residente.

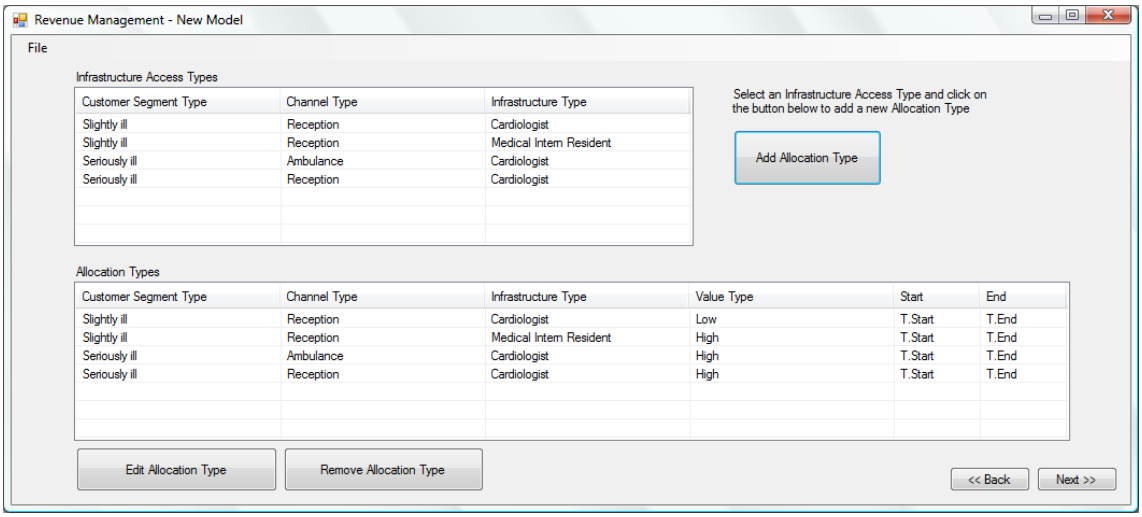


Figura 7-6. Ejemplo del Sector Sanitario: definición de los Allocation Types

A continuación, una vez hemos terminado de configurar el modelo de proceso, creamos una instancia del mismo que se utilizará para como punto de partida para el análisis de las necesidades de personal. Para ello, se le da un nombre a la instancia (Personnel Analysis, Análisis de Personal) y acto seguido se establece qué atributos son necesarios (figura 7-7).

En primer lugar, se requiere una distribución de llegadas para los pacientes de cada segmento, que denominaremos Arrival Distribution Data. En lo que respecta a los atributos de Infrastructure Set, se necesita uno que indique cuál es el número de médicos de cada tipo disponibles en la Unidad (Quantity). Por otra parte, el atributo Max Waiting List indica cuál es el número máximo de pacientes que pueden permanecer en espera de ser atendidos por un determinado tipo de médico.

Los atributos Value y Time almacenan la magnitud concreta de cada valor (High y Low) y cada tiempo (T.Start y T.End). Por otra parte, se introducen dos tipos de atributos de Infrastructure Access: los que hacen referencia a la probabilidad de paso de ciertos estados a otros y los destinados a guardar el tiempo que un acceso o asignación

debe permanecer en un determinado estado antes de pasar a otro. Finalmente, los atributos de Allocation sirven de criterio para determinar si se puede ofrecer atención al paciente y, en su caso, si recibe atención médica inmediatamente o pasa a lista de espera.

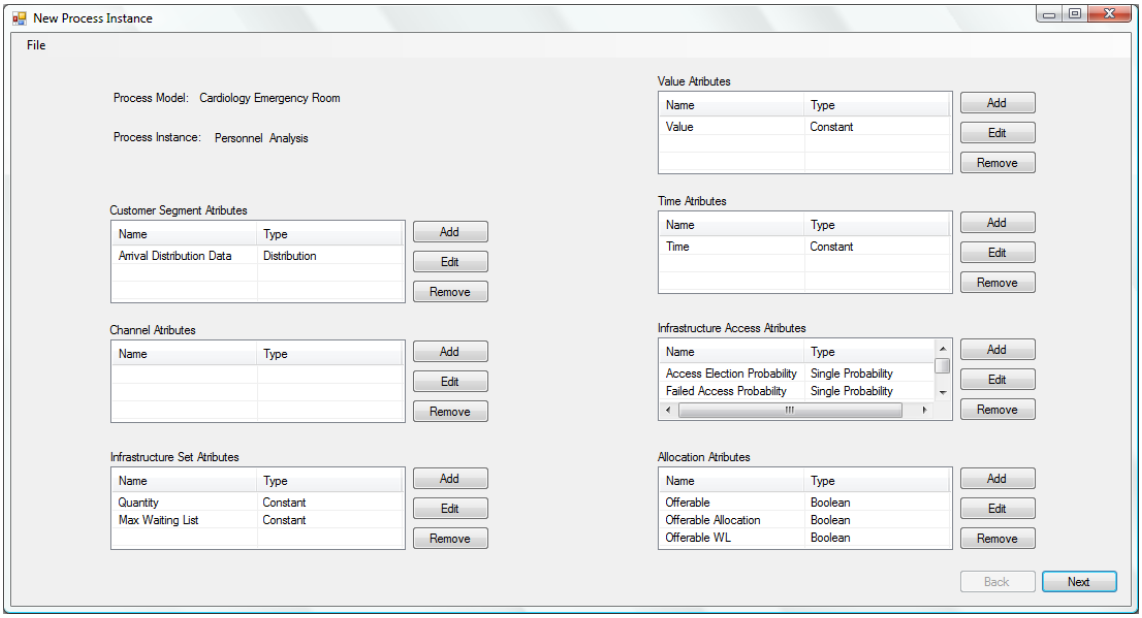
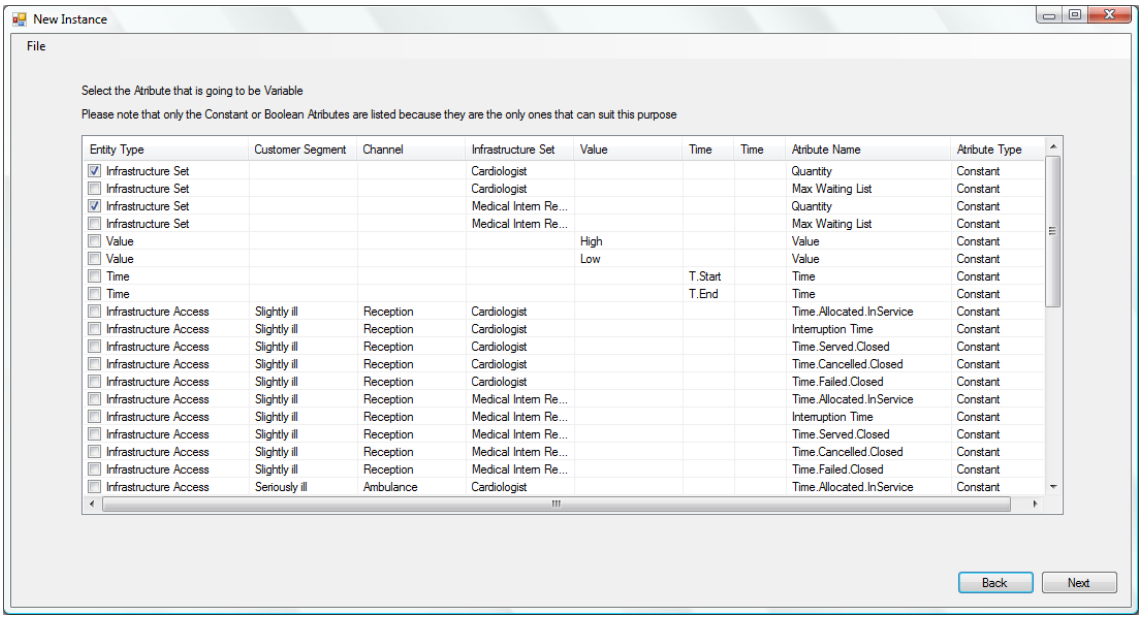


Figura 7-7. Ejemplo del Sector Sanitario: definición de Atributos

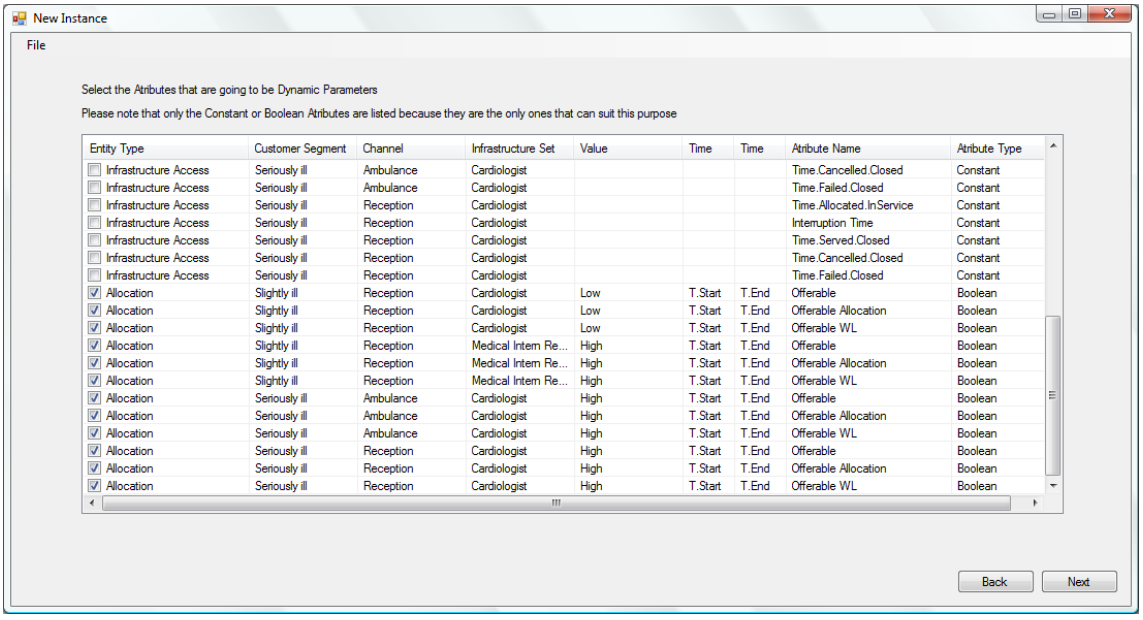
Una vez se pasa al siguiente formulario, se decide qué atributos van a actuar como variables, como puede apreciarse en la figura 7-8. Ya que en esta instancia se está planteando un problema de dimensionamiento óptimo de la plantilla de la Unidad, las dos variables serán las cantidades de médicos de cada tipo (cardiólogos y médicos internos residentes) que deben estar presentes en un turno (Quantity, atributo de Infrastructure Set).

A continuación, se designa a algunos de los atributos como parámetros dinámicos. Se elige como tales a todos los atributos de Allocation. Lo que tienen de particular estos atributos, como ya se explicó en el capítulo 4 del presente documento, es que sin ser variables independientes, carecen de un valor fijo, ya que éste depende de las circunstancias de una hipotética ejecución del proceso de negocio. Por ejemplo, el valor del atributo Offerable Allocation dependerá de si hay algún médico disponible de entre los pertenecientes al tipo que le ha sido asignado al paciente (valor verdadero, True) o no (valor falso, False). Esta disponibilidad fluctúa con el tiempo pero no es independiente como ocurre en una variable, sino que se trata de un atributo parametrizado y cuyo valor es dependiente. Esta configuración se presenta en la figura 7-9.



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Atribute Name	Atribute Type
☑ Infrastructure Set			Cardiologist				Quantity	Constant
☐ Infrastructure Set			Cardiologist				Max Waiting List	Constant
☑ Infrastructure Set			Medical Intern Re...				Quantity	Constant
☐ Infrastructure Set			Medical Intern Re...				Max Waiting List	Constant
☐ Value				High			Value	Constant
☐ Value				Low			Value	Constant
☐ Time					T.Start		Time	Constant
☐ Time					T.End		Time	Constant
☐ Infrastructure Access	Slightly ill	Reception	Cardiologist				Time.Allocated.InService	Constant
☐ Infrastructure Access	Slightly ill	Reception	Cardiologist				Interruption Time	Constant
☐ Infrastructure Access	Slightly ill	Reception	Cardiologist				Time.Served.Closed	Constant
☐ Infrastructure Access	Slightly ill	Reception	Cardiologist				Time.Cancelled.Closed	Constant
☐ Infrastructure Access	Slightly ill	Reception	Cardiologist				Time.Failed.Closed	Constant
☐ Infrastructure Access	Slightly ill	Reception	Medical Intern Re...				Time.Allocated.InService	Constant
☐ Infrastructure Access	Slightly ill	Reception	Medical Intern Re...				Interruption Time	Constant
☐ Infrastructure Access	Slightly ill	Reception	Medical Intern Re...				Time.Served.Closed	Constant
☐ Infrastructure Access	Slightly ill	Reception	Medical Intern Re...				Time.Cancelled.Closed	Constant
☐ Infrastructure Access	Slightly ill	Reception	Medical Intern Re...				Time.Failed.Closed	Constant
☐ Infrastructure Access	Seriously ill	Ambulance	Cardiologist				Time.Allocated.InService	Constant

Figura 7-8. Ejemplo del Sector Sanitario: definición de variables



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Atribute Name	Atribute Type
☐ Infrastructure Access	Seriously ill	Ambulance	Cardiologist				Time.Cancelled.Closed	Constant
☐ Infrastructure Access	Seriously ill	Ambulance	Cardiologist				Time.Failed.Closed	Constant
☐ Infrastructure Access	Seriously ill	Reception	Cardiologist				Time.Allocated.InService	Constant
☐ Infrastructure Access	Seriously ill	Reception	Cardiologist				Interruption Time	Constant
☐ Infrastructure Access	Seriously ill	Reception	Cardiologist				Time.Served.Closed	Constant
☐ Infrastructure Access	Seriously ill	Reception	Cardiologist				Time.Cancelled.Closed	Constant
☐ Infrastructure Access	Seriously ill	Reception	Cardiologist				Time.Failed.Closed	Constant
☑ Allocation	Slightly ill	Reception	Cardiologist	Low	T.Start	T.End	Offerable	Boolean
☑ Allocation	Slightly ill	Reception	Cardiologist	Low	T.Start	T.End	Offerable Allocation	Boolean
☑ Allocation	Slightly ill	Reception	Cardiologist	Low	T.Start	T.End	Offerable WL	Boolean
☑ Allocation	Slightly ill	Reception	Medical Intern Re...	High	T.Start	T.End	Offerable	Boolean
☑ Allocation	Slightly ill	Reception	Medical Intern Re...	High	T.Start	T.End	Offerable Allocation	Boolean
☑ Allocation	Slightly ill	Reception	Medical Intern Re...	High	T.Start	T.End	Offerable WL	Boolean
☑ Allocation	Seriously ill	Ambulance	Cardiologist	High	T.Start	T.End	Offerable	Boolean
☑ Allocation	Seriously ill	Ambulance	Cardiologist	High	T.Start	T.End	Offerable Allocation	Boolean
☑ Allocation	Seriously ill	Ambulance	Cardiologist	High	T.Start	T.End	Offerable WL	Boolean
☑ Allocation	Seriously ill	Reception	Cardiologist	High	T.Start	T.End	Offerable	Boolean
☑ Allocation	Seriously ill	Reception	Cardiologist	High	T.Start	T.End	Offerable Allocation	Boolean
☑ Allocation	Seriously ill	Reception	Cardiologist	High	T.Start	T.End	Offerable WL	Boolean

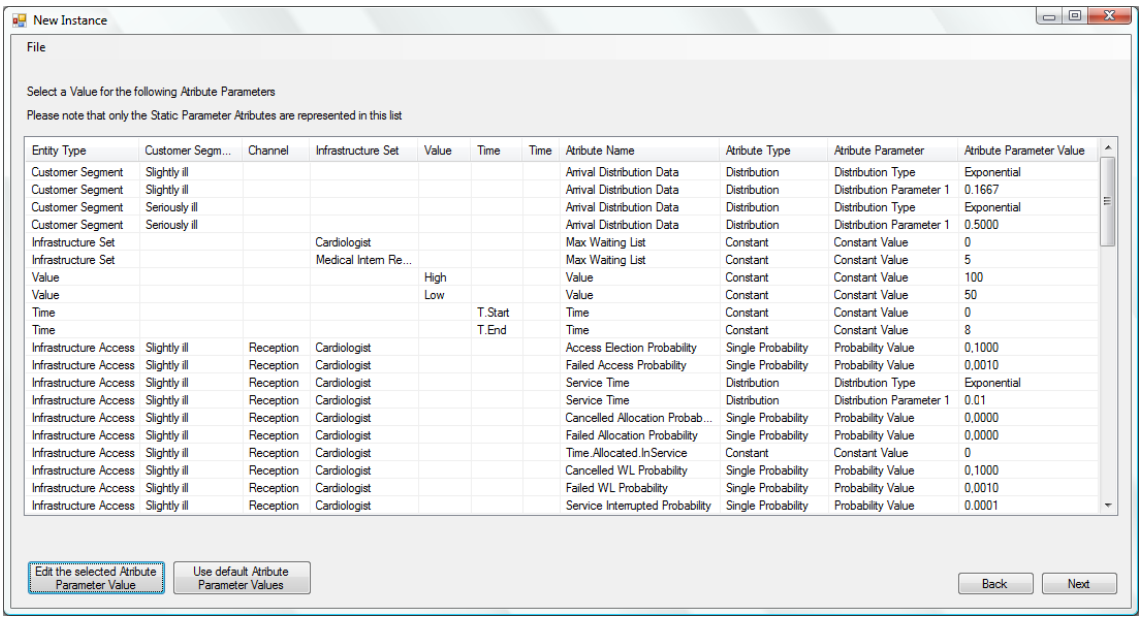
Figura 7-9. Ejemplo del Sector Sanitario: elección de parámetros dinámicos

Una vez se han seleccionado los atributos que actuarán como variables y como parámetros dinámicos, los atributos restantes se consideran parámetros estáticos. Como consecuencia, sus correspondientes parámetros de atributo adquieren un valor fijo en la instancia de proceso, a elección del usuario de la herramienta (figura 7-10).

En nuestro ejemplo, todos los valores que utiliza el analista provienen de los datos históricos de la Unidad. Tomando como unidad temporal la hora, se determina estadísticamente que la diferencia entre los tiempos de llegada de pacientes con

pronóstico leve se ajusta a una distribución exponencial de media 0.1667 horas, es decir, 10 minutos. Esto quiere decir que, de media, cada 10 minutos llega a la unidad un nuevo paciente con diagnóstico leve. La misma operación se realiza para los pacientes con diagnóstico de gravedad

En lo que respecta a las listas de espera, no se va a permitir que los pacientes de los cardiólogos entren en esta lista, con lo que si la función objetivo se elige adecuadamente (como se verá posteriormente), se va a forzar a nunca falte un cardiólogo cuando se le necesite. Por otro lado, como se va a estudiar lo que sucede en un turno de trabajo típico de ocho horas, se introduce que el valor temporal de T.Start es 0 y el de T.End, 8. Al resto de parámetros de atributos se les da el valor correspondiente siguiendo en función de los datos de los que se disponga. Quizá el único aspecto importante a tener en cuenta es que las probabilidades están expresadas en la unidad de tiempo. Por ejemplo, el hecho de que el parámetro de atributo Probability Value del atributo Cancelled WL Probability valga 0.1000 indica que un cliente asociado al Infrastructure Access del atributo tiene esa probabilidad por cada unidad de tiempo (en nuestro caso, por cada hora) que permanece en espera de marcharse y abandonar el hospital.



Entity Type	Customer Segm...	Channel	Infrastructure Set	Value	Time	Time	Atribute Name	Atribute Type	Atribute Parameter	Atribute Parameter Value
Customer Segment	Slightly ill						Arrival Distribution Data	Distribution	Distribution Type	Exponential
Customer Segment	Slightly ill						Arrival Distribution Data	Distribution	Distribution Parameter 1	0.1667
Customer Segment	Seriously ill						Arrival Distribution Data	Distribution	Distribution Type	Exponential
Customer Segment	Seriously ill						Arrival Distribution Data	Distribution	Distribution Parameter 1	0.5000
Infrastructure Set			Cardiologist				Max Waiting List	Constant	Constant Value	0
Infrastructure Set			Medical Intern Re...				Max Waiting List	Constant	Constant Value	5
Value				High			Value	Constant	Constant Value	100
Value				Low			Value	Constant	Constant Value	50
Time					T.Start		Time	Constant	Constant Value	0
Time					T.End		Time	Constant	Constant Value	8
Infrastructure Access	Slightly ill	Reception	Cardiologist				Access Election Probability	Single Probability	Probability Value	0,1000
Infrastructure Access	Slightly ill	Reception	Cardiologist				Failed Access Probability	Single Probability	Probability Value	0,0010
Infrastructure Access	Slightly ill	Reception	Cardiologist				Service Time	Distribution	Distribution Type	Exponential
Infrastructure Access	Slightly ill	Reception	Cardiologist				Service Time	Distribution	Distribution Parameter 1	0.01
Infrastructure Access	Slightly ill	Reception	Cardiologist				Cancelled Allocation Probab...	Single Probability	Probability Value	0,0000
Infrastructure Access	Slightly ill	Reception	Cardiologist				Failed Allocation Probability	Single Probability	Probability Value	0,0000
Infrastructure Access	Slightly ill	Reception	Cardiologist				Time.Allocated.InService	Constant	Constant Value	0
Infrastructure Access	Slightly ill	Reception	Cardiologist				Cancelled WL Probability	Single Probability	Probability Value	0,1000
Infrastructure Access	Slightly ill	Reception	Cardiologist				Failed WL Probability	Single Probability	Probability Value	0,0010
Infrastructure Access	Slightly ill	Reception	Cardiologist				Service Interrupted Probability	Single Probability	Probability Value	0.0001

Figura 7-10. Ejemplo del Sector Sanitario: elección de valores de los parámetros de atributo de los parámetros estáticos

El siguiente paso consiste en configurar los estados y cambios de estado por los que pasarán todos los accesos y asignaciones (figura 7-11). De entre los estados que ofrece el software como predefinidos, nos quedamos con todos excepto con dos, puesto que los pacientes, tratándose de una Unidad de Cardiología de Urgencias, no tendrán posibilidad de decidir si aceptan o rechazan al médico que les atiende. Además será

necesario añadir un par de cambios de estado que aseguren la continuidad del árbol de cambios de estado.

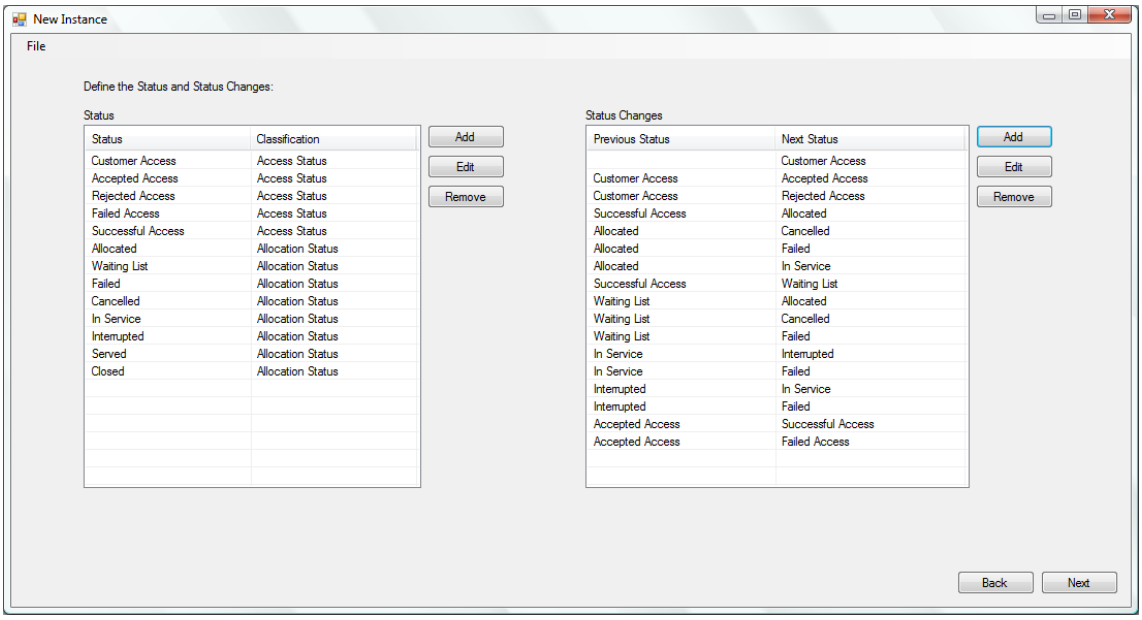
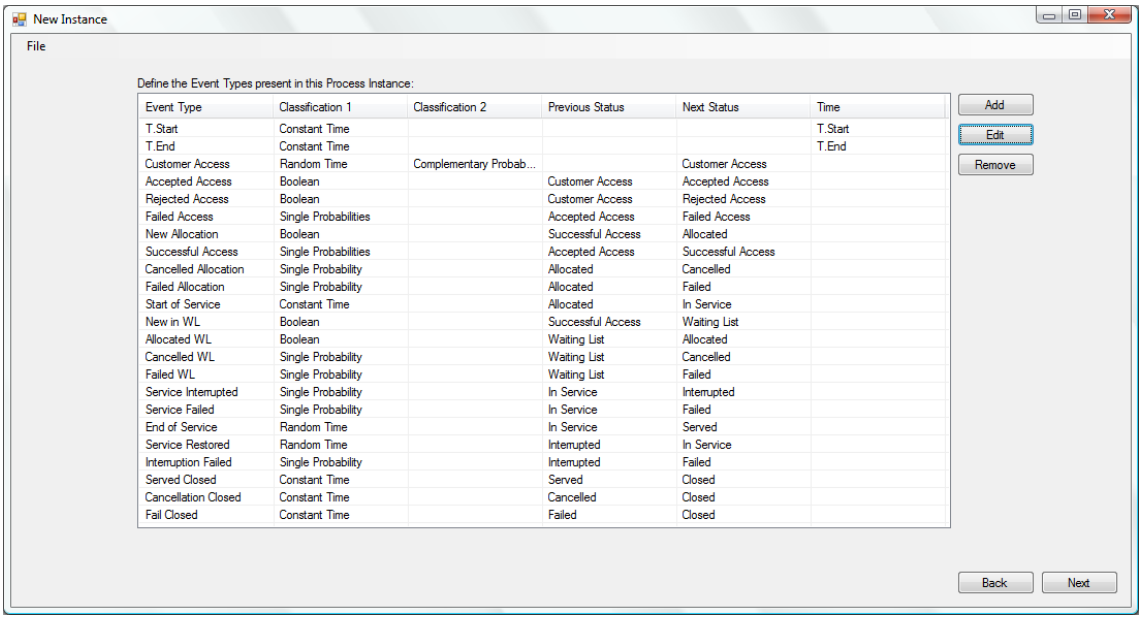


Figura 7-11. Ejemplo del Sector Sanitario: definición de estados y cambios de estado

La dinámica de la instancia de proceso, recogida en las figuras 7-11, 7-12 y 7-13, es la siguiente:

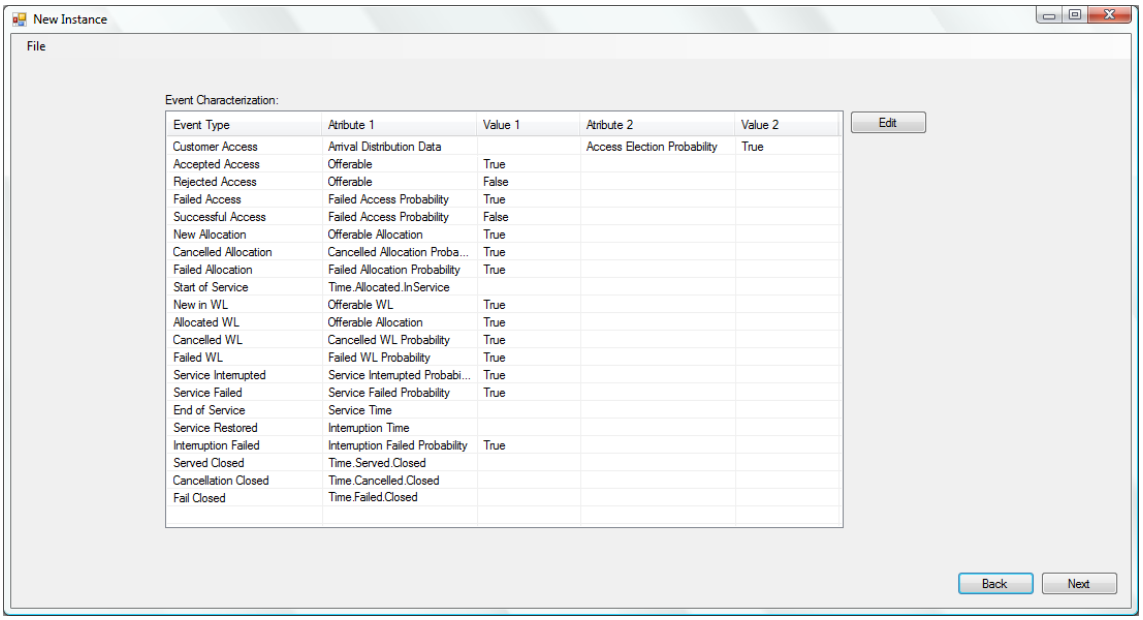
- Un paciente, con diagnóstico grave o leve, llega a la Unidad, bien sea en ambulancia o bien procedente de la recepción, para ser atendido por un cardiólogo o un médico interno residente. La distribución de llegadas de cada segmento determina cuándo llega un paciente de ese segmento. A continuación, se determina probabilísticamente, a través de los atributos denominados Access Election Probability, cuál es el canal y el conjunto de infraestructura (en este caso el tipo de médico) asociados al acceso de ese paciente (el evento relacionado es Customer Access).
- Posteriormente, en función de si hay capacidad disponible, es decir, si hay médicos libres para atenderle o bien puede pasar a lista de espera, se acepta (Accepted Access) o se rechaza el acceso (Rejected Access).
- Los accesos de pacientes aceptados son asignados a un médico concreto (New Allocation) o pasan a lista de espera (New in WL). Aquellos a los que se les asigna un médico, tras un intervalo de tiempo de una cierta duración, se procede a atender al paciente (Start of Service). No obstante, sería posible, con una determinada probabilidad, que en el intervalo de tiempo mencionado el paciente muriese (Failed Allocation) o abandonase el hospital por voluntad propia (Cancelled Allocation).



Event Type	Classification 1	Classification 2	Previous Status	Next Status	Time
T.Start	Constant Time				T.Start
T.End	Constant Time				T.End
Customer Access	Random Time	Complementary Probab...		Customer Access	
Accepted Access	Boolean		Customer Access	Accepted Access	
Rejected Access	Boolean		Customer Access	Rejected Access	
Failed Access	Single Probabilities		Accepted Access	Failed Access	
New Allocation	Boolean		Successful Access	Allocated	
Successful Access	Single Probabilities		Accepted Access	Successful Access	
Cancelled Allocation	Single Probability		Allocated	Cancelled	
Failed Allocation	Single Probability		Allocated	Failed	
Start of Service	Constant Time		Allocated	In Service	
New in WL	Boolean		Successful Access	Waiting List	
Allocated WL	Boolean		Waiting List	Allocated	
Cancelled WL	Single Probability		Waiting List	Cancelled	
Failed WL	Single Probability		Waiting List	Failed	
Service Interrupted	Single Probability		In Service	Interrupted	
Service Failed	Single Probability		In Service	Failed	
End of Service	Random Time		In Service	Served	
Service Restored	Random Time		Interrupted	In Service	
Interruption Failed	Single Probability		Interrupted	Failed	
Served Closed	Constant Time		Served	Closed	
Cancellation Closed	Constant Time		Cancelled	Closed	
Fail Closed	Constant Time		Failed	Closed	

Figura 7-12. Ejemplo del Sector Sanitario: definición de tipos de eventos

- Los pacientes que permanecen en lista de espera son asignados a un médico concreto cuando alguno perteneciente al tipo que debe atenderlos queda libre (Allocated WL). También puede que mueran durante la espera (Failed WL) o que decidan no esperar más y se marchen del hospital (Cancelled WL).
- Mientras un paciente es atendido, puede producirse una interrupción (Service Interrupted) o puede morir (Service Failed). Si se produce una interrupción, ésta durará un cierto tiempo, tras el cual el paciente sigue siendo atendido (Service Restored), si no se produce su muerte en ese periodo (Interruption Failed). Si es atendido con éxito, finaliza el servicio (Service Restored)
- Finalmente, todos los accesos y asignaciones fallidos, cancelados o servidos se consideran cerrados.



Event Type	Atribute 1	Value 1	Atribute 2	Value 2
Customer Access	Arrival Distribution Data		Access Election Probability	True
Accepted Access	Offerable	True		
Rejected Access	Offerable	False		
Failed Access	Failed Access Probability	True		
Successful Access	Failed Access Probability	False		
New Allocation	Offerable Allocation	True		
Cancelled Allocation	Cancelled Allocation Proba...	True		
Failed Allocation	Failed Allocation Probability	True		
Start of Service	Time.Allocated.InService			
New in WL	Offerable WL	True		
Allocated WL	Offerable Allocation	True		
Cancelled WL	Cancelled WL Probability	True		
Failed WL	Failed WL Probability	True		
Service Interrupted	Service Interrupted Probabi...	True		
Service Failed	Service Failed Probability	True		
End of Service	Service Time			
Service Restored	Interruption Time			
Interruption Failed	Interruption Failed Probability	True		
Served Closed	Time.Served.Closed			
Cancellation Closed	Time.Cancelled.Closed			
Fail Closed	Time.Failed.Closed			

Figura 7-13. Ejemplo del Sector Sanitario: caracterización de eventos

La caracterización de eventos de la figura 7-13 muestra los valores que deben adquirir los parámetros de atributo de ciertos atributos para que ocurran los tipos de eventos a los que están asociados.

El último paso consiste en definir la función objetivo de la instancia de proceso. La función es de maximización y tiene en cuenta varios factores importantes a la hora de dimensionar la plantilla de un hospital.

En primer lugar, hay que asegurar que se presta un servicio de calidad y no se compromete en exceso la salud ni el bienestar de los pacientes. Esta consideración tiene dos vertientes. Por un lado, existe un término en la función objetivo que penaliza con gran severidad las muertes de pacientes en lista de espera (Deaths WL). Por otra parte, otro término penaliza, si bien en menor medida, el tiempo que los pacientes permanecen en lista de espera (Time WL).

Secundariamente, se intenta que, siempre y cuando la calidad del servicio sea adecuada, los costes de personal sean los menores posibles, por lo que hay dos términos que reflejan el número de médicos que están de servicio que, recordemos, son las variables del problema (Number of MIRs y Number of Cardiologists).

Finalmente, se desea que se atienda con éxito al mayor número posible de pacientes y de la manera más adecuada, por lo que un término tiene en cuenta el beneficio social proveniente de la atención recibida por los enfermos. Ese término es una combinación del número de pacientes atendidos para cada valor (Served) y el propio valor (Value).

Las figuras 7-14, 7-15, 7-16 y 7-17 muestran los detalles de la definición de la función objetivo.

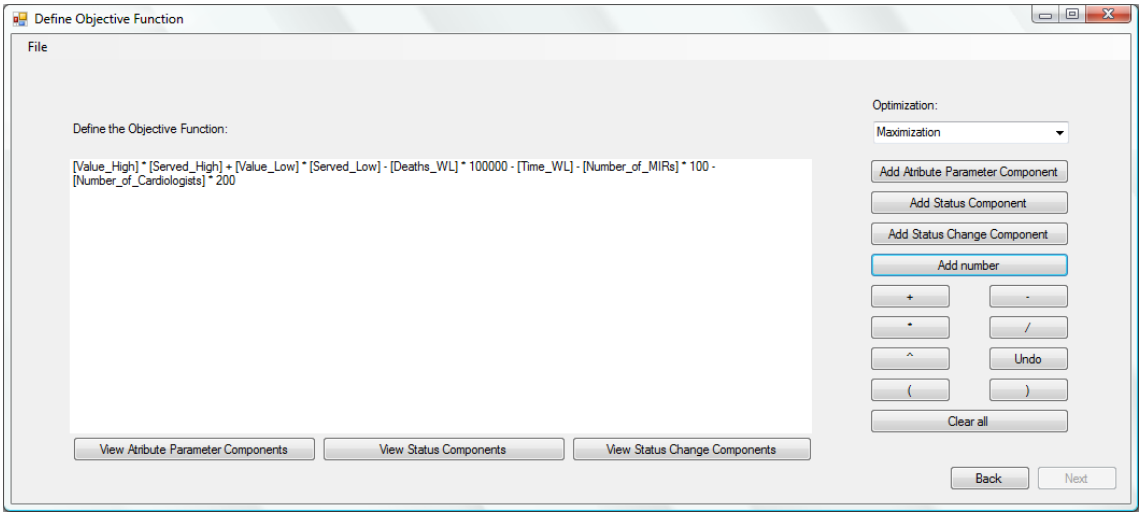


Figura 7-14. Ejemplo del Sector Sanitario: definición de la función objetivo

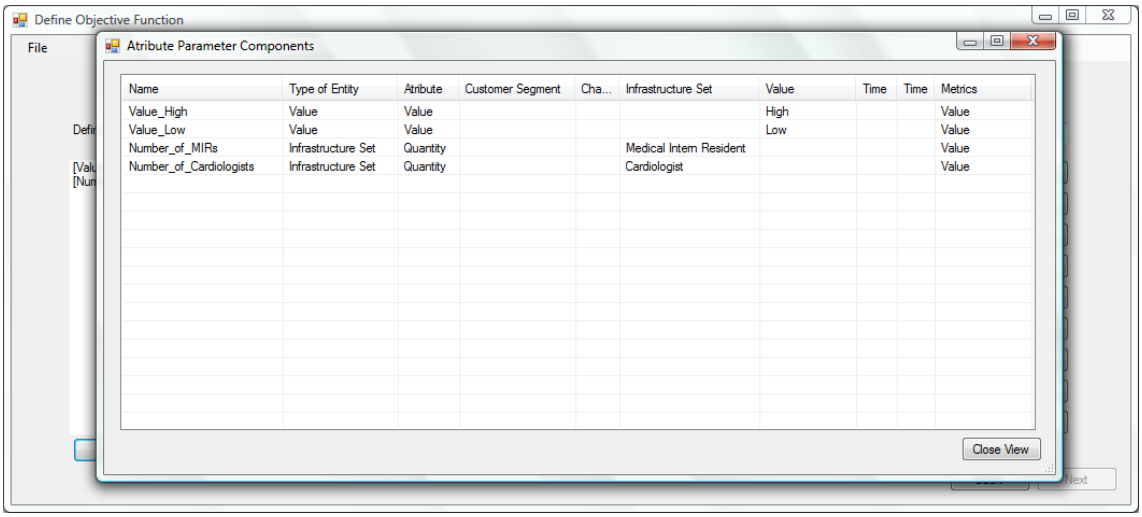


Figura 7-15. Ejemplo del Sector Sanitario: Attribute Parameter Components añadidos

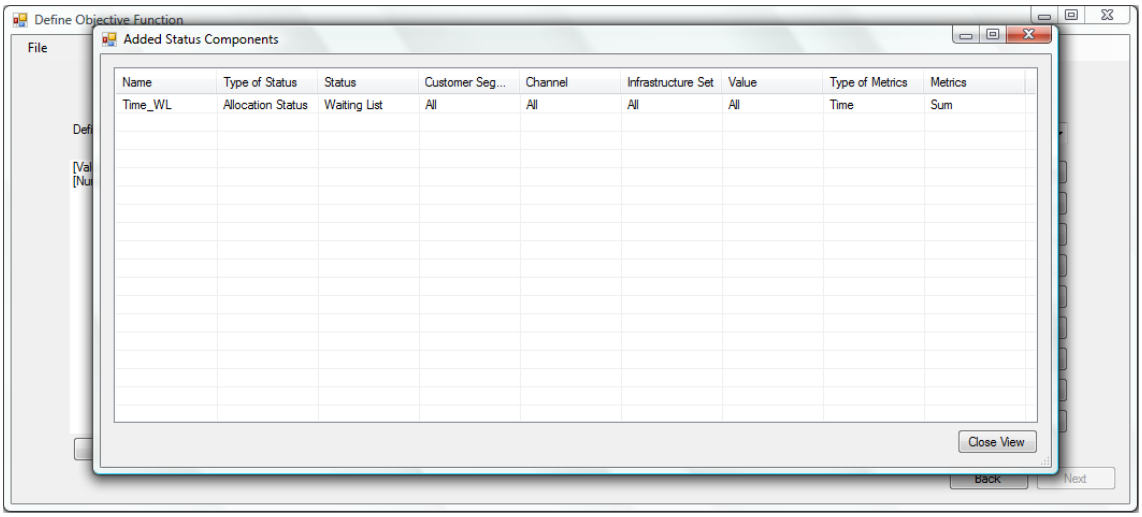


Figura 7-16. Ejemplo del Sector Sanitario: Status Component añadido

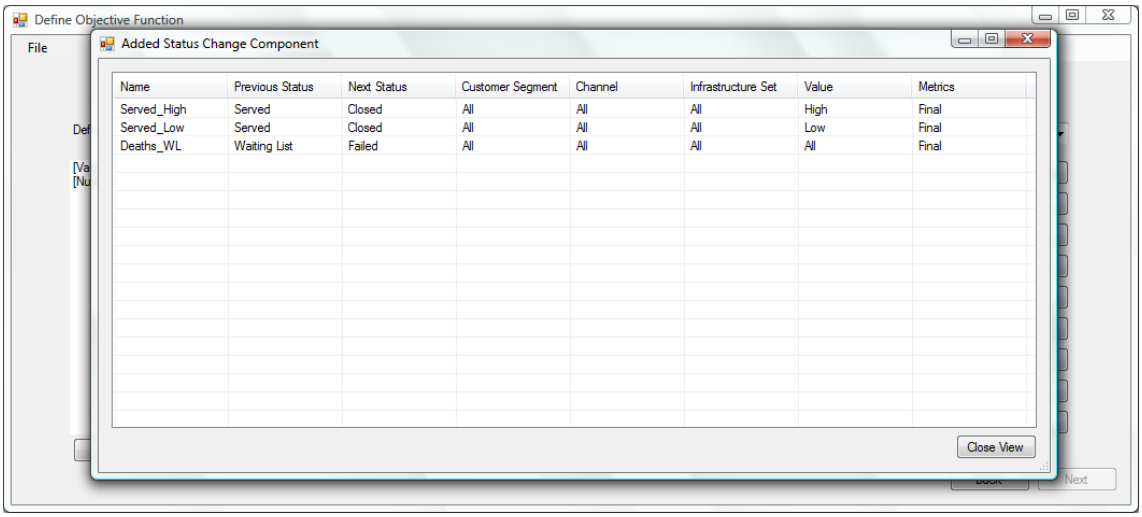


Figura 7-17. Ejemplo del Sector Sanitario: Status Change Components añadidos

8

CONCLUSIONES

8.1 CONCLUSIONES

Se partía con el objetivo de desarrollar una herramienta informática fácilmente expandible que permitiese modelar de forma flexible, rápida e intuitiva problemas de asignación de infraestructuras a clientes, pertenecientes a un segmento o agrupación, que acceden por distintos canales para realizar una reserva.

Se ha ido exponiendo a lo largo de todo el documento el proceso de creación de la herramienta, tal y como se anunció en la estructura. En primer lugar, se explicaron los fundamentos conceptuales del modelo de partida, sobre el que se asientan las bases del proyecto, se anunciaron las líneas generales de diseño y se mostraron las herramientas de desarrollo que cubrieron la necesidad de disponer de una solución específica para el modelo conceptual (UML), el modelo de datos (IDEF1X) y la interfaz gráfica de usuario (Visual Basic).

A continuación, como se adelantó en el diseño de la herramienta, se dedicó un capítulo a cada uno de los cuatro módulos del proyecto. El primero de ellos, el módulo de definición del Modelo de Proceso, está vinculado a los dos primeros niveles jerárquicos del modelo de partida, el nivel de metamodelo y el nivel de modelo. En cambio, los otros tres están relacionados con el tercer nivel jerárquico, correspondiente a la instancia de proceso de negocio. En cada uno de estos capítulos se describió el diseño conceptual, el modelo de datos, el diseño de la aplicación y la implementación del módulo correspondiente.

Finalmente, el capítulo de ejemplo permite mostrar lo intuitiva, rápida y, sobre todo, flexible que es la definición de un modelo y una instancia de proceso con la asistencia de la herramienta, teniendo en cuenta que estas características formaban parte de los requisitos que se le solicitaban a la herramienta al inicio del proyecto.

Retomando la cuestión de las herramientas de desarrollo con más detalle, la combinación de los diagramas de clase UML para el diseño conceptual con los diagramas IDEF1X para ilustrar el modelo de datos facilita enormemente el proceso posterior de programación. En concreto, IDEF1X es una herramienta excelente para el programador poco acostumbrado al diseño de bases de datos, pues con una serie de conceptos básicos y muy claros, se pueden modelar las tablas necesarias así como sus relaciones, facilitando su representación e implementación, ya que el gráfico de relaciones de MS Access resulta algo confuso. Por otra parte, la elección de Visual Basic como lenguaje de programación cumplió con las expectativas. A pesar de que fue necesario iniciar su aprendizaje desde cero, no ha sido difícil interiorizar su funcionamiento y aplicarlo con éxito para obtener un software robusto y ejecutable en un entorno tan extendido como MS Windows.

En cuanto al detalle concreto del desarrollo del módulo de definición de un Modelo de Proceso, la parte conceptual del Process Model cuya implementación ha supuesto una mayor complejidad ha sido la relacionada con los Generic Times, principalmente la comprobación del no solapamiento de Allocation Types derivados de un mismo Infrastructure Access Types, que requiere tanto un algoritmo de

comprobación como la definición previa de las matrices de restricciones explícitas e implícitas, de tipo igual y menor o igual.

Profundizando en el desarrollo del módulo de definición de Atributos, la parte del diseño conceptual cuya implementación ha supuesto una mayor dificultad y tiempo de programación corresponde a la evaluación de los Attribute Parameters, principalmente en lo que respecta al guardado y la carga de los datos en la base de datos. También fue el formulario más complejo de diseñar, tanto en lo que respecta a la forma de mostrar la información como a la manera de seleccionar una serie de valores por defecto. La introducción de una serie atributos predefinidos y de valores por defecto para los parámetros de atributo pretende facilitar al usuario la tarea de configurar una instancia de proceso. De hecho, no pueden entenderse algunos de los atributos que se han introducido sin conocer el contenido del módulo de definición de Estados. No obstante, la aplicación no pierde ni un ápice de flexibilidad, de tal manera que aquellos usuarios que pretendan configurar instancias con atributos completamente distintos a los que aquí se han considerado como importantes pueden hacerlo sin ningún tipo de problema. Así mismo, si los valores de los Attribute Parameters se consideran poco adecuados, ya sean todos o bien sólo una parte, pueden modificarse fácilmente.

En lo que atañe al detalle del módulo de definición de Estados, con el fin de no complicar en exceso el proceso de definición de las entidades que forman parte de dicho módulo, se ha desechado la posibilidad de que los distintos Infrastructure Access puedan llegar a adquirir Status distintos, que la secuencia de los mismos sea diferente o que los tipos de eventos que motivan los cambios de estado difieran entre unos Infrastructure Access y otros. Esta decisión simplifica notablemente la aplicación y mejora su experiencia de uso, renunciando a una funcionalidad que no es demasiado útil ni de uso muy habitual.

Los Status, Status Changes, Event Types y Event Characterization cargados en el software por defecto surgen como respuesta a la preocupación durante la fase de diseño ante el excesivo tiempo que podría ser necesario para un usuario cualquiera configurar una instancia de proceso desde cero. Se pretende facilitar el uso de la aplicación sin renunciar a la posibilidad de que el software permita la definición de una gran variedad de problemas.

No obstante, para no perder el enfoque global del problema, cabe recordar que el modelo de la dinámica de una instancia de proceso, también conocido como modelo de estados, surge como requisito previo a la introducción de un modelo de función objetivo, ya que algunos términos susceptibles de aparecer en la misma no se pueden concebir con una visión estática del Process Instance.

Sopesando con más detenimiento el desarrollo del modelo de función objetivo, se puede afirmar que éste se beneficia del modelo de la dinámica de la instancia de proceso de negocio para abordar la definición de una amplia gama de problemas de gestión del ingreso con asignación de infraestructuras. Gracias al enfoque elegido para abordar este proceso, que proporciona bastante libertad al usuario, la meta de optimización puede no ser exclusivamente económica, para tener en cuenta otros

aspectos relacionados con la calidad del servicio: como pueden ser la minimización de tiempos de espera o de clientes desatendidos. Estos factores y otros del mismo tipo han tenido gran importancia desde siempre en ciertos sectores, como puede ser el sanitario, y tienen una relevancia cada vez mayor en cualquier sector económico, debido al incremento de la competencia y a la saturación de los mercados.

La funcionalidad que permite al usuario definir sumatorios de componentes dentro de la función objetivo se ha incluido en el software como fruto de la preocupación ante el excesivo tiempo que podría ser necesario para definir una función objetivo típica, no necesariamente muy compleja, en una instancia de proceso en la que existiese un número considerable de entidades.

Finalmente, cabe plantearse hasta qué punto se ha cumplido con los requisitos del proyecto. En lo que respecta a dotar al software de una cimentación conceptual, se ha logrado configurar un modelo sólido, bien fundamentado y sencillo de expandir hacia el siguiente nivel jerárquico, el de ejecución, con lo que existe la posibilidad de que esta herramienta llegue a formar parte de un DSS en un futuro. Además, se ha respetado la arquitectura de dos elementos reflejada en los requisitos, con lo que se facilita esa eventual expansión de funcionalidades. En cuanto al papel de las herramientas de desarrollo como soluciones en materia de modelado conceptual, de datos y de interfaz de usuario, el resultado es satisfactorio. Por último, hay que preguntarse cómo de flexible, rápida e intuitiva es la definición de problemas con el software. Su flexibilidad es indudable, pues permite plantear un amplísimo rango de problemas de asignación de infraestructuras muy diferentes entre sí. Sin embargo, precisamente esta flexibilidad ha puesto en compromiso los otros dos requisitos, que no alcanzan el mismo grado de consecución, si bien se encuentran en un rango aceptable. Se ha identificado dos posibles desarrollos futuros que mejorarían la rapidez y lo intuitivo de la aplicación. Se trata de la definición y carga de plantillas y la representación gráfica del árbol de cambios de estado, respectivamente.

8.2 FUTUROS DESARROLLOS

Tal y como hemos anunciado en la sección anterior, un buen método para mejorar lo intuitivo de la aplicación, sería actuar sobre el formulario de definición de cambios de estado de los accesos y asignaciones, puesto que la apariencia actual, en forma de lista, no es sencilla de interpretar. Podría implementarse una interfaz gráfica que mostrase un árbol de estados, con cada nodo representando un estado. Incluso los eventos que producen un cambio de estado podrían situarse, ya en el siguiente formulario, sobre las flechas que unen los distintos nodos. La apariencia final sería muy parecida a la de los diagramas de estado UML que se han utilizado en la explicación del módulo de definición de Estados.

El otro requisito cuyo grado de consecución mejoraría con un desarrollo futuro sería el de la rapidez de definición de problemas. Podría concebirse la creación y carga de plantillas que permitiesen al usuario, por ejemplo, saltarse la definición de la

dinámica de la instancia, contar con una dinámica estándar o incluso con varias plantillas para dar valores estándar a los parámetros de atributo de los parámetros estáticos.

Po último, destacar que el desarrollo futuro más evidente y además el más relevante de todos es, como se ha anunciado en la sección previa, el diseño e implementación de un simulador que, actuando sobre el problema definido por la herramienta informática que nos ocupa, sea capaz de simular la ejecución del proceso de negocio, de tal forma que se puedan obtener los valores de las variables que optimizan la función objetivo. La modularidad de la aplicación y la arquitectura de dos capas facilitan enormemente una eventual integración del simulador en la herramienta como un módulo más de ésta. Recordemos que el simulador no se ha incluido en el presente proyecto por un motivo de alcance, puesto que nuestro enfoque se ha situado sobre el planteamientos del subconjunto de problemas de asignación de infraestructuras que implican el acceso de un cliente, perteneciente a un segmento, con el fin de reservar una infraestructura a través de un cierto canal; y no sobre la resolución de este tipo de problemas.

9

BIBLIOGRAFÍA

BIBLIOGRAFÍA

BRUCE, Thomas A. *Designing Quality Databases with IDEF1X Information Models*. 1ª Edición. EEUU: Dorset House Publishing Company, 1991. 584 p. ISBN: 978-0932633187.

DATE, C. J. *SQL and Relational Theory: How to Write Accurate SQL Code*. 1ª Edición. EEUU: O'Reilly Media, 2009. 428 p. ISBN: 978-0596523060.

FOWLER, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* 3ª Edición. EEUU: Addison Wesley Professional, 2003. 208 p. ISBN: 978-0321193681.

GADDIS, Tony; IRVINE, Kip R. *Starting Out with Visual Basic 2010*. 5ª Edición. EEUU: Addison Wesley, 2010. 896 p. ISBN: 978-0136113409.

GUTIÉRREZ, Miguel; DURÁN, Alfonso. *Generic Model Base Design for Decision Support Systems in Revenue Management: Applications to Hotel and Health Care Sectors*. 2011.

ROB, Peter; CORONEL, Carlos. *Database Systems: Design, Implementation and Management.* 7ª Edición. EEUU: Course Technology, 2006. 668 p. ISBN: 978-1418835934.

SILVER, Mark S. *Systems That Support Decision Makers: Description and Analysis*. 1ª Edición. EEUU: Wiley, 1991. 272 p. ISBN: 978-0471919681.

SPRAGUE, Ralph H; CARLSON, Eric. *Building Effective Decision Support Systems*. 1ª Edición. EEUU: Prentice Hall, 1982. 304 p. ISBN: 978-0130862150.

THOMSEN, Carsten. *Programación de Bases de Datos con Visual Basic .NET*. 1ª Edición. España: Inforbook's Ediciones, 2002. 479 p. ISBN: 84-95318-89-X.

TURBAN, Efraim; ARONSON, Jay E.; LIANG, Ting-Peng. *Decision Support Systems and Intelligent Systems.* 7ª Edición. EEUU: Prentice Hall, 2004. 960 p. ISBN: 978-0130461063.

YEOMAN, Ian; McMAHON-BEATTIE, Una. *Revenue Management and Pricing: Case Studies and Applications*. 1ª Edición. EEUU: Thomson Learning, 2004. 216 p. ISBN: 1-84480-062-8.

ZENI, Richard H. *Improved Forecast Accuracy in Airline Revenue Management by Unconstraining Demand Estimates from Censored Data*. 1ª Edición. EEUU: Dissertation.com, 2001. 276 p. ISBN: 978-1581121414.